

REUSE CAN MAKE YOUR PROGRAMMING EASIER

BEA WebLogic

DEVELOPER'S JOURNAL

AUGUST 2003 - Volume:2 Issue:8

weblogicdevelopersjournal.com

Web Services Edge ^{WEST} 2003

web services **EDGE**
conference & expo



SEPT. 30 - OCT. 2, 2003
Santa Clara, CA

CONFERENCE PROGRAM PAGE 33

FROM THE EDITOR
Pattern Matching
by Sean Rhody
page 5

DEVELOPER
RELATIONS
WebLogic Support
by Apurb Kumar
page 46

ASK THE EXPERT
Keeping It Simple
by Lewis Cirne
page 48

NEWS &
DEVELOPMENTS
page 50

RETAILERS PLEASE DISPLAY
UNTIL SEPTEMBER 30, 2003
\$8.99US \$9.99CAN



0 09281 03424 7

SYS-CON
MEDIA



WALTER HURST 6

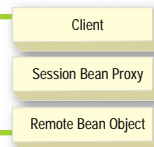
DESIGN PATTERNS: **J2EE Framework for Services** A different approach may add benefits



12

Petter Graff

DESIGN PATTERNS: **Improved Performance with the EJBSelect Data Aggregation Pattern** Increased performance and better management



18

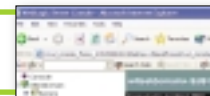
Tom Purcell

DESIGN PATTERNS: **J2EE Design Patterns** Keep a few handy in your tool belt

22

Alan Baumgarten

DESIGN PATTERNS: **The Promise of Patterns** An idea that should not be missed



26

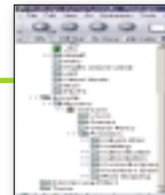
Bill Willis

ADMINISTRATION: **Extending the Admin Console for Your Custom Security Provider** Two ways to add to your functionality

30

Mike Kennedy

FROM THE OFFICE OF THE CTO: **The Language, the Server, and the JVM** There's movement deep within the stack



40

Benjamin Renaud

PRODUCT REVIEW: **BEA WebLogic Integration 8.1 from BEA Systems** Moving ahead of its competitors

42

Jason Snyder

TRANSACTION MANAGEMENT: **Freedom, Disasters, and Getting Something for Nothing**



44

Peter Holditch

Hewlett-Packard

<http://devresource.hp.com/wldj.html>

Confluent Software

<http://www.confluentsoftware.com>



BY SEAN RHODY
MANAGING EDITOR

Pattern Matching

When I first graduated (well, actually, the second time) I had an offer from a company for a programming job. They were going to hire me, contingent upon my passing the IBM Programmer's Aptitude Test. So one day I drove my college junk heap an hour out of my way to take this test. I had to get an "A" to get the job. I was nervous, but once I got the test, I realized I needn't have worried – it was all about patterns.

It's both a relief and a disappointment in the programming business that nothing is really new. We keep solving the same old problems, sometimes in new ways, or with new tools. That's disappointing, but at the same time it allows us to focus on repeatability. It also lets us focus on the business side of our careers (i.e., the things that pay the bills) by developing a certain amount of productivity. In a sense, as knowledge workers (that's such a nasty term) we're like highly educated assembly-line workers – paid to do the same things over and over again. Our added value (and the justification for our billing rates or salaries) is that unlike an assembly line worker, who is normally given a set pattern, it's our job to identify patterns in business processes or their technical implementations, and take advantage of them for maximum efficiency.

Nowhere in programming are patterns and best practices more important than in the J2EE world. In the beginning, there were no J2EE patterns, and people did all sorts of strange things with their code. It compiled, it even ran, but at the end of the day it performed poorly.

You should, of course be familiar with the idea of patterns. If you're not, put this down, go buy *Design Patterns* by Gamma et al., and study up, then pick this issue up again.

Patterns are everywhere. The JavaBean and the EJB approach are based on patterns such as the getter/setter method of defining an attribute. Or the use of deployment descriptors.

But these aren't enough to ensure performance, which is why the folks at Sun put together the J2EE Blueprints. These are a group of patterns that help to optimize the performance of a J2EE application. The Fast Lane Reader pattern, for example, helps reduce the overhead of transferring read-only (or read mostly) data to user displays by avoiding the instantiation of entity beans. Because when you send the data across the EJB container to the Web container, you don't send the EJB, you send its data.

Even these patterns need proper care and feeding. It's easy to go overboard and make everything session beans (or even JSP pages), but this doesn't accomplish the goals of a partitioned architecture, nor does it leverage the transaction management and storage mechanisms of the EJB container. There's a balance to pattern usage, and it's very important to consider that balance before acting.

This issue presents some patterns for J2EE use. Properly designed, a J2EE application can scale to virtually any level of client support. Improperly designed, you can choke a user community to death with a system that cannot be saved even by throwing hardware at it. It's up to you to learn what patterns exist, and not just what patterns, but when it's appropriate to use them. Do so and you're on your way to the transition from programmer to architect. And to making things count for your business or client.

I got an "A" on the aptitude test, but I took a different job that paid better and offered more skill development in modern languages and systems. Did they really think I wanted to learn COBOL? Yikes! 🍌

AUTHOR BIO...

Sean Rhody is the managing editor of *WebLogic Developer's Journal* and editor-in-chief of *Web Services Journal*. He is a respected industry expert and a consultant with a leading consulting services company.

CONTACT: sean@sys-con.com

EDITORIAL ADVISORY BOARD

TOM ANGIUCCI, LEWIS CIRNE, STUART HALLOWAY,
KEVIN JONES, TYLER JEWELL, WAYNE LESLEY LUND,
SEAN RHODY, BLAKE STONE

FOUNDING EDITOR

PETER ZADROZNY

EDITOR-IN-CHIEF

JASON WESTRA

EDITORIAL DIRECTOR

JEREMY GEELAN

MANAGING EDITOR

SEAN RHODY

PRODUCT REVIEW EDITOR

JASON SNYDER

EXECUTIVE EDITOR

GAIL SCHULTZ

EDITOR

NANCY VALENTINE

ASSOCIATE EDITORS

JAMIE MATUSOW, JEAN CASSIDY

ASSISTANT EDITOR

JENNIFER VAN WINCKEL

WRITERS IN THIS ISSUE

ALAN BAUMGARTEN, LEWIS CIRNE, PETER GRAFF,
PETER HOLDITCH, WALTER HURST, MIKE KENNEDY,
APURB KUMAR, TOM PURCELL, BENJAMIN RENAUD,
SEAN RHODY, JASON SNYDER, BILL WILLIS

SUBSCRIPTIONS

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department.

SUBSCRIPTION HOTLINE:

888-303-5282

Cover Price: \$8.99/Issue

Domestic: \$149/YR (12 Issues)

Canada/Mexico: \$169/YR

Overseas: \$179/YR

(U.S. Banks or Money Orders)

PRESIDENT AND CEO

FUAT A. KIRCAALI

VP, BUSINESS DEVELOPMENT

GRISHA DAVIDA

SENIOR VP, SALES & MARKETING

CARMEN GONZALEZ

PRODUCTION CONSULTANT

JIM MORGAN

ART DIRECTOR

ALEX BOTERO

ASSOCIATE ART DIRECTORS

LOUIS F. CUFFARI • RICHARD SILVERBERG

ASSISTANT ART DIRECTOR

TAMMI BEATTY

VP, SALES & MARKETING

MILES SILVERMAN

ADVERTISING SALES DIRECTOR

ROBYN FORMA

DIRECTOR OF SALES AND MARKETING

MEGAN RING-MUSSA

ADVERTISING SALES MANAGER

ALISA CATALANO

ASSOCIATE SALES MANAGERS

CARRIE GEBERT • KRISTIN KUHNLE

PRESIDENT, SYS-CON EVENTS

GRISHA DAVIDA

CONFERENCE MANAGER

MICHAEL LYNCH

FINANCIAL ANALYST

JOAN LAROSE

ACCOUNTS RECEIVABLE

KERRI VON ACHEN

ACCOUNTS PAYABLE

BETTY WHITE

VP, INFORMATION SYSTEMS

ROBERT DIAMOND

WEB DESIGNERS

STEPHEN KILMURRAY • CHRISTOPHER CROCE

ONLINE EDITOR

LIN GOETZ

CIRCULATION SERVICE COORDINATORS

NIKI PANAGOPOULOS • SHELLA DICKERSON

EDNA EARLE RUSSELL

JDJ STORE MANAGER

RACHEL MCGOURAN

EDITORIAL OFFICES

SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 822-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
BEA WebLogic Developer's Journal (ISSN# 1535-9581)
is published monthly (12 times a year)
Postmaster: Send Address Changes to
BEA WEBLOGIC DEVELOPER'S JOURNAL,
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

WORLDWIDE NEWSSTAND DISTRIBUTION
Curtis Circulation Company, New Milford, NJ



Wily Technology

www.wilytech.com



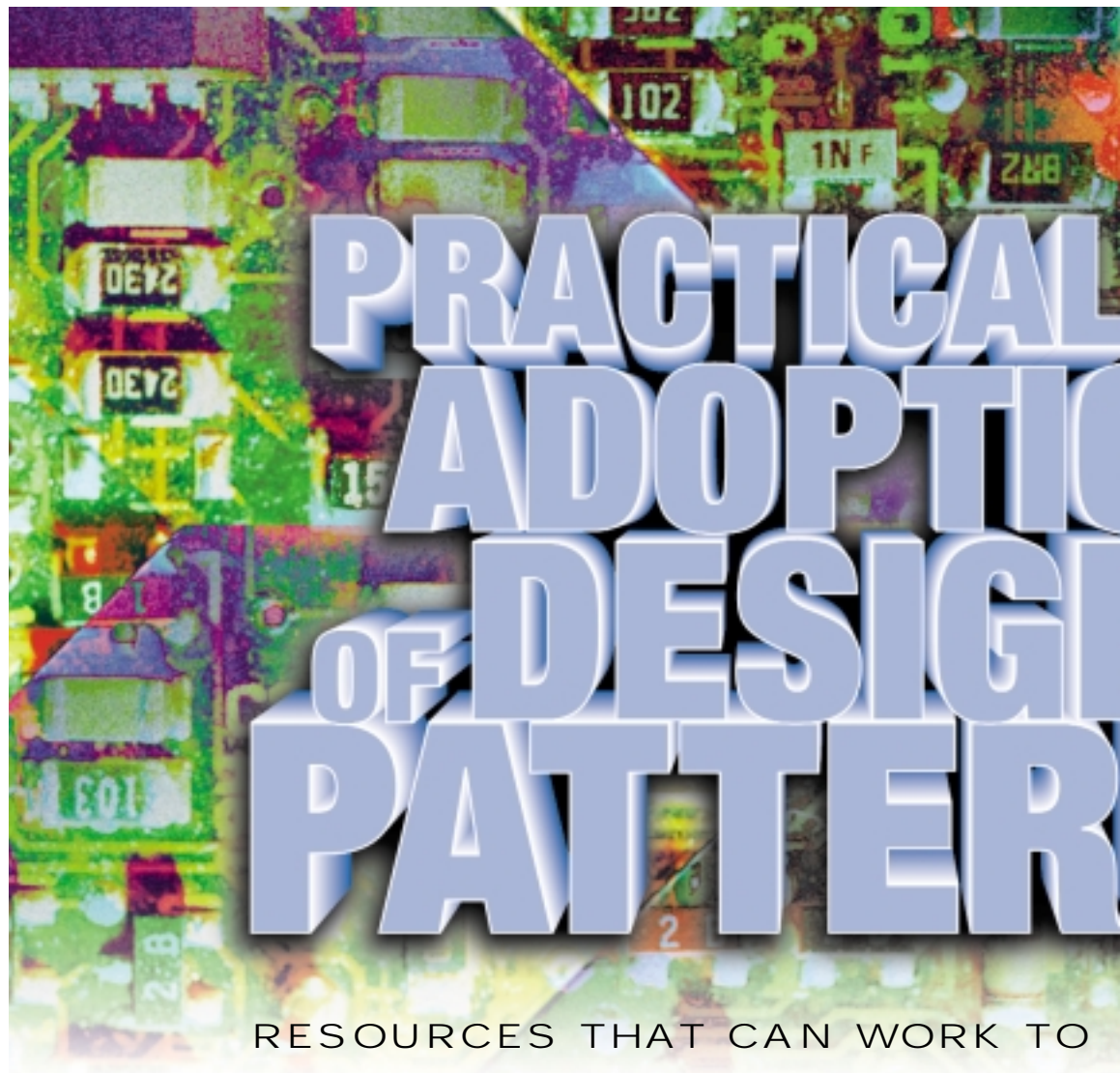
BY WALTER HURST

AUTHOR BIO...

Walter Hurst is the founder and CTO of Wakesoft. He has been working with advanced technologies for over 10 years. Before founding Wakesoft, Walter was an independent consultant and used early versions of Wakesoft technology to deliver Internet solutions to clients. Walter received a BS in computer engineering from the University of Michigan.

CONTACT...

walter@wakesoft.com



RESOURCES THAT CAN WORK TO

Often there is a distinction between concepts and how those concepts are applied in the real world. Design patterns are no exception.

Design patterns are everywhere. It's hard to read a technology publication or Web site without seeing some reference to design patterns. By now you have probably read (or at least paged through) some of the design pattern books, such as *Core J2EE Design Patterns* or *Design Patterns* by the Gang of Four. At this point you are possibly asking some questions about design patterns. How will design patterns help me? Are they a silver bullet? Are there any problems with design patterns? Don't I get design patterns from my integrated development environment (IDE)?

All of these are good questions to ask as you go through the process of adopting design patterns. Often there is a distinction between concepts and how those concepts are applied in the real world. Design patterns are no exception. In this article we'll look at the real world adoption of design patterns. This information should help you make

the right decisions about successfully adopting design patterns in your project.

Quick Overview

Design patterns provide a way to share experiences so that a community can benefit and not continually reinvent the wheel. Design patterns typically capture a problem description, the context of the problem, a proposed solution to the problem, and any anticipated consequences of using the solution. In order to have the broadest applicability (and therefore usefulness to the largest audience), design patterns are often abstracted from exact details that are situation dependent. This abstractness results in some required interpretation to apply the design pattern to the case at hand. This is an important detail: even though design patterns are a great way to share expertise, it is often not trivial to correctly apply that expertise.

The concept of design patterns originated in the construction industry. Architects (the ones that design buildings, not computer systems) realized they needed to share ideas about correct



design techniques. These ideas were captured in design patterns that allowed the architect community to benefit from shared experiences and lessons learned. Design patterns made the leap from construction to computer systems in the late '80s. Object-oriented (OO) philosophies were gaining popularity and design patterns were a way to educate the new OO followers on best practices.

The book that launched design patterns into the limelight is *Design Patterns: Elements of Reusable Object-Oriented Software* by Richard Gamma, et al (the authors are commonly referred to as the Gang of Four [GoF]). As design patterns have grown in popularity, the number of flavors they come in has also ballooned in what could be described as the "Ben and Jerry" effect. For those of you not familiar with the popular ice cream brand, Ben and Jerry's is a purveyor of fine ice cream products that come in every imaginable combination of ingredients (including some you would never have ever imagined). So it is with

design patterns, which originated with the general OO design patterns from the GoF book but now include design patterns that are specific for development languages, application servers, industry verticals, and so on.

Categorizing Design Patterns

Design patterns are often grouped together based on some common characteristics. The GoF book divided design patterns into three categories: Creational, Behavioral, and Structural. Design patterns for J2EE are often divided into Presentation Tier, Business Logic Tier, and Integration Tier. This grouping may make it easier to describe common details that all the design patterns share or simply make it easier to catalog and discover design patterns.

For our discussion of the practical adoption of design patterns, we need to divide design patterns into two categories: broad exposure and isolated use. This division is based on the visibility of the design pattern to the designers and developers of the application and the dependency of multiple parts of the application on the design pattern.

Broad exposure design patterns are distinguished by affecting the designs and development of more than just one team member or more than one area of the application. Some qualities of this type of design pattern are:

- Adopting it has the downstream effect of many classes being created that depend on the design pattern.
- Different parts of the application are aware of the use of the design pattern.
- The decision to use this design pattern cannot be easily undone.

Some examples of this type of design pattern are the Model-View-Controller (MVC) pattern, the Value Object J2EE pattern, and the Data Access Object (DAO) J2EE pattern.

Isolated use refers to design patterns for which the use of the design pattern is a hidden detail. Some qualities of this type of design pattern are:

- The design pattern does not impact the work of other team members or other parts of the application.
- The decision to use the design pattern could easily be changed with minimal impact.

Some examples of this type of design pattern are the Singleton GoF pattern or the Intercepting Filter J2EE pattern.

Dividing design patterns into these categories provides a quick approach for understanding the scope of the design pattern. Understanding scope makes it easy to assess the impact of the design pattern. Is it

a design pattern that can be used and discarded? Is it a design pattern that once adopted will impact the design of the application? Is it a design pattern that affects multiple parts of my application and possibly other applications? Understanding these impacts up front provides guidance for adopting the design pattern.

Design Pattern Adoption AntiPatterns

As design patterns grew in popularity, another type of patterns emerged called AntiPatterns. Whereas design patterns offer expert knowledge on repeatable best approaches, AntiPatterns describe often repeated behavior that should be avoided. AntiPatterns prove there is as much to be learned from people who do things wrong as those who do things right.

In this section, we'll look at AntiPatterns that apply to the adoption of design patterns. Understanding these AntiPatterns should help you avoid the pitfalls of design pattern adoption. Like design patterns, the AntiPatterns here may be entirely new to you, in which case they provide some foresight; or they may be situations you are somewhat familiar with, in which case they can add color to your experiences and let you know that you're not alone. If you are interested in reading more AntiPatterns, see the resource list at the end of this article.

AntiPattern Checklist

Design Patterns? Yes, We Have All of Them

Problem: Determining which design patterns to include on a project.

Applies To: Both broad exposure and isolated use design patterns.

Context: A developer has been introduced to design patterns and wants to use them on a project.

Forces: Two forces are usually the source for this AntiPattern. One is the desire of the developer to improve the project by including best practices from design patterns. The other is that innate curiosity drives the developer to use the project for research on design patterns.

Supposed Solution: All known design patterns are used on the project. A checklist is formed from the design pattern book and the motivating goal is to be able to check off all of the design patterns.

Resulting Context: The project team and the delivered application suffer from the unnatural introduction of too many design patterns. This results in designs and development that are overly complex. This unnecessary complexity can impact the development effort in terms of the amount of work that

has to be done, the ability of the development team to understand what is happening, and the actual performance and functional correctness of the application.

Design Rationale: Design patterns are a great source of expert knowledge. If using them is good, then using all of them is mistakenly thought of as better.

Real Solution: The description of a design pattern includes the context for which the pattern is targeted. Thought must be given to ensure that the design pattern is a match for the project. Second, design patterns are often incorporated not from someone reading a design pattern book and asking “where can I use this design pattern?” but from someone looking for a solution to a problem they have discovered.

Implementation per Developer/Project AntiPattern

(Also known as: Design pattern xyz?
Yeah, we have 10 of those.)

Problem: Controlling the implementation of design patterns within a project and across projects.

Applies To: Both broad exposure and isolated use design patterns benefit from addressing this situation. However, broad exposure design patterns should definitely have controlled implementations.

Context: The development team is incorporating design patterns into the project. The team consists of many experienced developers who understand when to use design patterns. The correct design patterns are being used. If multiple projects are involved, there is no design pattern-implementation sharing between projects.

Forces: There are tight deadlines and the team is very efficient individually. Team efficiency might be impacted if they have to reuse implementations. It is assumed that since they are experts, their implementation will be good anyway. In the case of multiple projects, the cross-team communication and code sharing either is not considered or is dismissed as a potential impact on schedules.

Supposed Solution: The team can include and implement the design patterns they need as individuals.

Resulting Context: Even though the correct design patterns were used, they were implemented in many different ways. There are incompatibilities between the implementations that restrict integration and reuse efforts. More time and effort than necessary will have to be spent maintaining, debugging, and extending the various implementations. Eventually the various implementations will have to be consolidated.

Design Rationale: A team of experts

should be allowed to work alone. Including design patterns is good enough, there is no reason to share implementations.

Real Solution: The development team should coordinate their use of design patterns. Sharing a common implementation of a design pattern can lower costs in the future, but more important, it should result in cross-developer compatibility today. If necessary, this sharing could be limited to design patterns that fall into the broad exposure category we discussed earlier. Reusing implementations is also valuable across projects, especially if they are expected to integrate in the future.

Role of IDEs in Design Pattern Adoption

IDEs continue to evolve and offer more functionality. Originally IDEs consisted of an editing environment and some debugging facilities. Now they often include a design environment, auditing facilities, integration with configuration management systems, and more. As IDEs expand their reach, their role in implementing design patterns needs to be identified. Of course, design patterns are implemented in a development language and IDEs can be used for editing the source code. But what other roles can IDEs play?

Some IDEs have pull-down menus that allow you to select design patterns to include in your application. While this can expedite the inclusion of a design pattern, it might just result in writing bad code faster. There are a couple of factors to remember in evaluating this feature.

One is that design patterns describe problems in the abstract and require some interpretation to arrive at the correct implementation. However, they often include a “sample implementation” and it is this sample class structure that the IDEs insert into your application. This is probably not the implementation that you need and just dropping it into your application will introduce more confusion and require more editing and refactoring than just putting thought into your own original implementation.

Another problem with the IDE drag-and-drop design pattern approach is the two AntiPatterns we already discussed. Expediting the implementation of design patterns is more likely to result in rampant inclusion of design patterns and multiple versions of the same design pattern than it is to solve any problems.

The challenge with design patterns isn't just in getting a quick implementation, but making sure you are using a correct imple-

mentation, and ideally an implementation that already exists in your organization.

BEA WebLogic Workshop 8.1 and Design Patterns

Since you are probably a BEA customer if you are reading this article, you may be wondering how the new BEA WebLogic Workshop 8.1 impacts your design pattern considerations. First of all, WebLogic Workshop is an IDE so the previous section on IDEs applies. The two additional aspects of Workshop that are interesting for this discussion are Controls and pre-implemented design patterns.

WebLogic Workshop Controls are a way to package functionality so that it can be easily included in your application using the Workshop IDE. The packaging includes the visual elements required for the IDE, the runtime behavior, the configuration required, and more. How do Controls impact your inclusion of design patterns? Remember the earlier segregation of design patterns into isolated use and broad exposure? Design patterns that fall into the isolated use category may be candidates to be packaged as Workshop Controls. Packaging the design pattern as a Control would allow other users of the Workshop IDE to share an implementation, avoiding the Implementation per Developer/Project AntiPattern.

You may be wondering why the broad exposure category cannot be implemented as Controls. The reason is that the broad exposure design patterns result in the creation of many other classes or in some dependence with other applications. This situation does not lend itself to the plug-and-play aspect of Controls. Adoption of the broad exposure design patterns should be done with much thought and consideration, and once adopted cannot be undone easily. These requirements do not match with the WebLogic Workshop Control goals.

WebLogic Workshop also comes with many pre-implemented design patterns, such as Pageflow and the user interface structure. From within Workshop, you can create JSPs and define Pageflows to control the navigation between the pages of your Web application. Under the hood, WebLogic Workshop uses the popular Apache Struts presentation layer framework. This aspect of Workshop (which uses Struts) provides a Model-View-Controller (MVC) design pattern implementation, which means you won't have to create your own MVC implementation. Workshop includes other functionality that will prob-

Quest Software

<http://java.quest.com/jcsv/wldj>

ably replace your own design pattern implementations. While it is nice to get some of these design patterns implemented out-of-the-box, you should validate that you are comfortable with not only the implementation but with any dependency on WebLogic Workshop that the implementation creates.

Three Steps for Successful Design Pattern Adoption

So how do you merge the adoption of design patterns with the tight deadlines, scaled back budgets, and limited team resources that exist in many companies today? The following are three steps to follow to make your design pattern experience successful.

Strong Communication and Education

Many organizations have technical leadership, possibly formalized as an architect or informally as the recognized experts. These leaders should promote open communication around design pattern adoption and should educate the teams about specific design patterns. The communication should be across development teams and projects to proactively prevent adoption silos and multiple unique implementations (remember the Implementation per Developer/Project AntiPattern). The education can take the form of informal internal lunch-and-learns, formalized internal classes, or maybe even sending a few people to external training. Any of these education options should promote correct applications of design patterns. If only a limited audience can attend the training, the best candidates are ones who feel comfortable educating their peers upon return.

Mentored Design Pattern Adoption

Design patterns can be used for the benefit of the project but they are also easy to misuse to the detriment of the application. Their use should be encouraged but their adoption should be reviewed and verified. Design patterns can be included during both design and development. In either case, design patterns' use should be identified and verified by the reviewer. Also during this review, situations may be encountered where additional design patterns may be applicable that weren't originally included. Even if official reviews are not done in your environment, this step could be accomplished through peer reviews or team discussions. The reviewers in this step should be either part of the leadership team or in open communication with them.

Mentoring adoption is especially critical for the broad exposure category of design patterns. These are the design patterns that have more associated risk because they will create dependencies. These dependencies may be within a class of objects, such as data access objects (DAOs) that only work within the scope of a broader DAO design pattern implementation, or across application boundaries, such as use of the Value Object design pattern to communicate data between applications and application layers. These design patterns are also ones that may have been implemented by somebody else on the project or by a different project, and that implementation should be reused versus creating another unique implementation.

Reuse Implementations, Not Just Design Patterns

While there is a certain satisfaction that comes in creating your own implementation of a design pattern, there is more benefit to the team and company when reuse takes place at the code level and not just the design idea level. The initial benefit design patterns provide to an organization is improved implementations. The real goal, however, should be to reuse implementations. Reusing implementations will result in: a) other reusable classes (that depend on the common implementation); b) lower development time and costs; c) lower maintenance time and costs; and d) easier integration both within and between applications.

This reuse is more important (and sometimes essential) for the broad exposure category. These design patterns are the ones that create external dependencies (where integration would benefit from a common implementation) or result in a whole library of custom classes (that would be reusable if there was a common foundation). The isolated use category would benefit from reuse but they might be more difficult to reuse if they were strongly customized to their specific situation.

Something you may ask yourself is, "If reuse is better, why don't design patterns come with a reusable implementation?" We touched on this when we talked about how design patterns are provided for the benefit of the largest audience. Achieving this goal of broad applicability would be difficult, if not impossible, if it came with a predetermined implementation. However, once the design patterns are applied to a particular problem domain or technical infrastructure, the resulting implementa-

tion may be reusable within that environment.

Design Patterns Through Architecture

It may seem a daunting task to understand how design patterns apply to your situation, build a quality implementation, and then encourage the reuse of that implementation. One approach to this is the introduction of application architecture into your environment. Application architecture provides the structure required for your application so that the development team can focus on the domain logic of the application. This includes implemented design patterns. Instead of reusing design pattern concepts or individual implementations, the architecture can be reused across projects and applications. This shared common implementation ensures compatibility and provides a lower cost alternative to developing and maintaining many different implementations. The compatibility provides the required technical foundation for reuse efforts to be possible. There is not enough space here to go into the other valuable qualities of architecture, such as runtime monitoring and management, configurable application logic, and adaptable behavior. You can learn more about architecture from the Carnegie Mellon Software Engineering Institute (www.sei.cmu.edu/ata/ata_init.html).

Summary

Design patterns are an amazing resource that should be used to your advantage. While design patterns provide reusable concepts, the challenges come in determining which design patterns to use and working towards a reusable implementation. By understanding some of the risks involved with design pattern adoption, you can hopefully avoid them as you continue to learn and implement more design patterns.

Following the steps outlined in this article will put a process in place that can be used to promote the successful adoption of design patterns on your team and within your organization.

Resources

- Brown, William J.; Malveau, Raphael C.; McCormick, Hays W. "Skip"; Mowbray, Thomas J. (1998). *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley & Sons
- *AntiPatterns*: www.antipatterns.com
- Wiki site: <http://c2.com/cgi/wiki?AntiPattern>

BEA Systems

<http://dev2dev.bea.com/useworkshop>



J2EE FRAMEWORK FOR SERVICES



BY PETER GRAFF

AUTHOR BIO...

Petter Graff is vice president of InferData Corporation. He has more than 20 years of experience building object-oriented software solutions. For the last 10 years he has been teaching and consulting on enterprise architectures for Fortune 500 companies worldwide.

CONTACT...

pgraff@inferdata.com

A DIFFERENT APPROACH MAY ADD SUBSTANTIAL BENEFITS

This article presents a lightweight framework for building a service-oriented architecture (SOA) on top of J2EE.

The architecture is a subset of a framework that we have successfully applied in various J2EE applications. The framework is built from a combination of commonly known design patterns (GoF = Gang of Four; j2eeep = J2EE Core Patterns):

- Factory (GoF)
- Proxy (GoF)
- Service Locator (j2eeep)
- Business Delegate (j2eeep)
- Service Facade (j2eeep)

At first glance, the framework may seem to require too many artifacts, but we will justify these as we go along. We will also see that most of the artifacts can be generated.

What Is Service-Oriented Architecture?

SOA is a framework for building IT solutions in the form of services. A service encapsulates some functionality exposed through one or more well-documented interfaces. The interfaces may be published and discoverable through repositories.

Could We Use Web Services?

A popular technique for communication, publishing, and discovery of services is the use of Web services and UDDI. Web services work well as an interapplication framework where the service interfaces provide only coarse-grained methods. However, Web services are too resource intense for intra-application communication. To leverage the idea of services inside a J2EE application, we need a framework with much less overhead.

Example

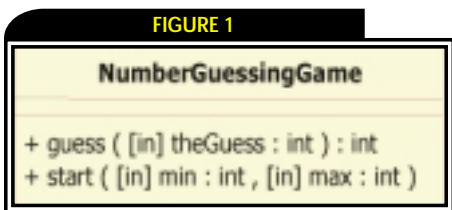
To illustrate the framework, we will create a very simple service that implements a number guessing game.

The number guessing game has two methods:

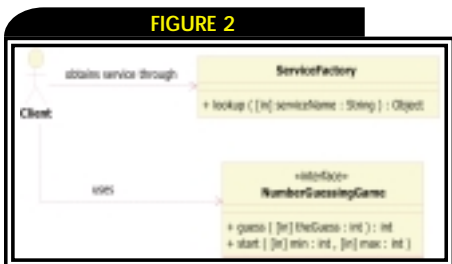
- The method **start** starts (or restarts) the game and sets the upper and lower boundary for possible guesses. The service generates a random secret number that it stores internally.
- The method **guess** is called when a client wants to guess the value of the secret number. The method returns the value - 1 if the guess is too low, 0 if the guess is correct, and 1 if the guess is too high.

To implement this simple service, we create the following artifacts:

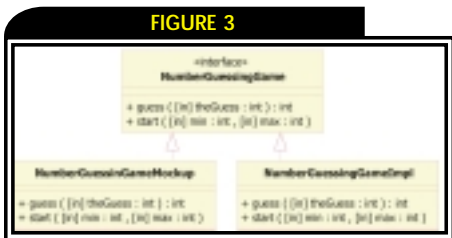
- **Service Interface:** Declaration of the interface for the service
- **Service Factory:** Used by service clients and session beans to instantiate services
- **Session Bean Proxy:** Client-side implementation of the service interface that forwards to a session bean
- **Business Bean:** The “real” service implementation
- **Mockup Bean:** A “fake” service implementation



The NumberGuessingGame interface



The client view of the NumberGuessingGame service



Implementation provided by the implementer

- **Session Bean:** A standard EJB session bean enabling enterprise-strength declaration of services (security, transactions, remote communication, etc.)

The Client's View

One goal of the framework is to make it easy for clients to obtain and use services. A client could be a Web component (JSP, servlet, or Struts action), a stand-alone application, or a business component running inside the EJB tier.

In our framework, the clients obtain services through a service factory. The service factory returns an object implementing an interface that the client expects. This shields the clients from the complexity of J2EE (JNDI lookup, narrowing, etc.). The following code fragment shows how a client obtains the number guessing game service:

```
NumberGuessingGame game = (NumberGuessingGame)
ServiceFactory.lookup("NumberGuessingGame");
```

With the clients' independence of J2EE artifacts now achieved, it is significantly easier to change the underlying implementation. This allows us to post-optimize the implementation of the number guessing game without affecting the client.

The Implementer's View

Ideally, the implementer of the business logic should also be shielded from the details of the EJB technology. The framework enables the business logic to be implemented as a standard JavaBean (see Figure 3).

The implementer would typically provide two implementations of the service interface. The first implementation is a

simple mockup that allows its clients to develop their logic without waiting for the real implementation to be complete. For this simple game, such a mockup is not justified, but it is very useful in enterprise applications where the development of the service may take a long time. A mockup reduces the risk of integration problems, especially when several teams concurrently implement services (see Figure 4).

A simple mockup implementation of the guess method could generate a random number from -1 to 1:

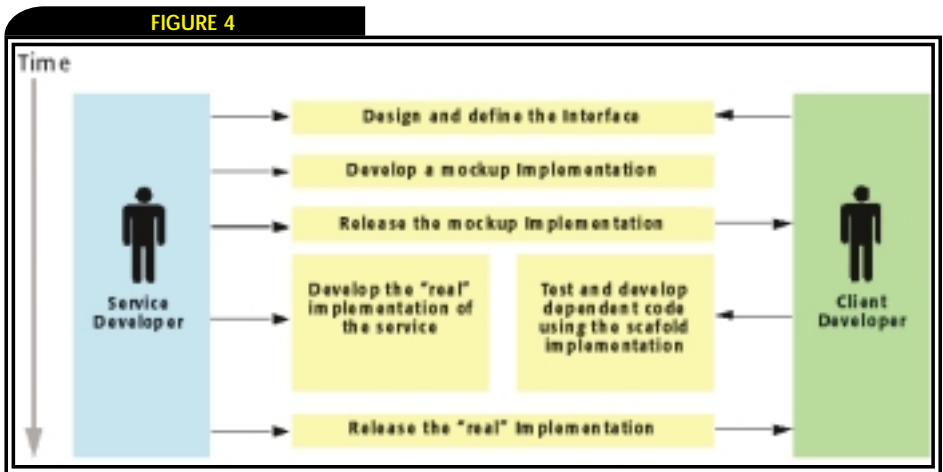
```
class NumberGuessingGameMockup implements
NumberGuessingGame {
public NumberGuessingGameMockup () {}
public void start(int min, int max) {}
public int guess(int aGuess) throws
TechnicalException {
return (((int)(Math.random() * 3.0)) -
1);
}
}
```

Later we would follow up with the real implementation shown in Listing 1.

The J2EE Connector Implementation

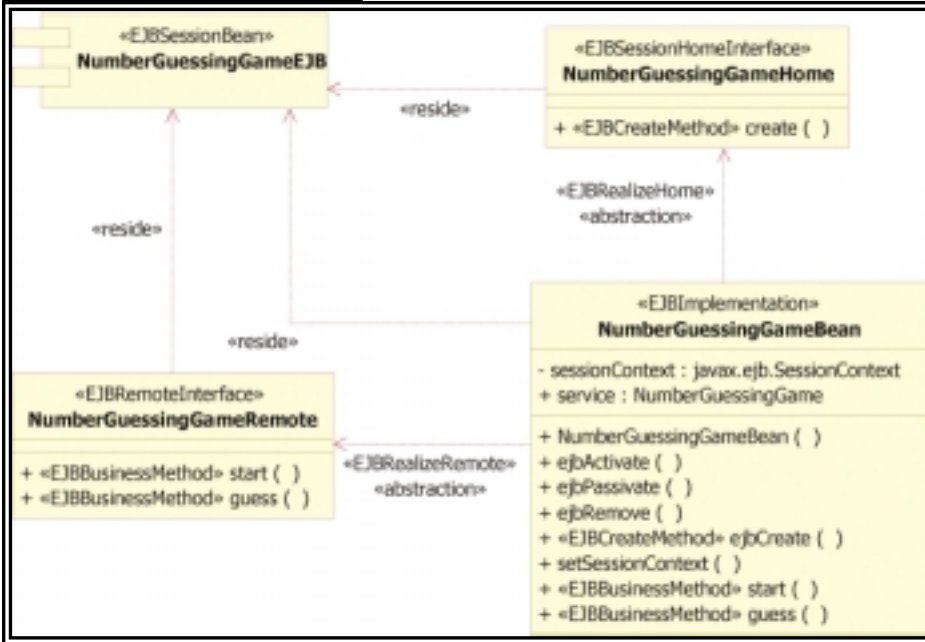
To enable a remote client to communicate with the business logic, we must provide an implementation of a proxy that has the service interface, but forwards to the business logic. This requires several artifacts in J2EE.

A proper port into the EJB container is a session bean, which provides server-side activation of the business logic. It receives remote business requests from clients, looks up the business logic bean, and forwards the business requests to it. The ses-



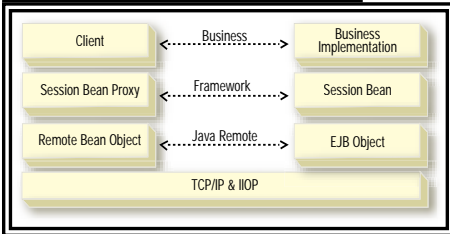
Process for service development

FIGURE 5



Implementation of the session bean

FIGURE 6



Framework layers

session bean is implemented according to standard EJB rules (see Figure 5).

Notice that the session bean has a reference to a service implementing the NumberGuessingGame interface. When the session bean is activated, it uses a server-side service factory to create the service:

```

void ejbCreate() {
    this.service =
        (NumberGuessingGame)
        ServiceFactory.lookup("NumberGuessingGame");
}
    
```

When a business method is invoked, the session bean simply forwards the request to the service:

```

int guess(int theGuess) throws
    TechnicalException {
    return this.service.guess( theGuess );
}
    
```

To provide the client view we discussed

earlier, we must also provide a client-side implementation that hides the use of the session bean from the client. We call this implementation a session bean proxy. In the core J2EE patterns, this implementation corresponds to the business delegate pattern.

The session bean proxy created for the client acts quite similarly to the session bean. Upon creation, it accesses JNDI to look up the home and creates the remote proxy from the home (see Listing 2).

The implementation of business methods on the session bean proxy forwards requests to the remote object. If a remote exception occurs, the exception is rethrown as a technical exception. For example:

```

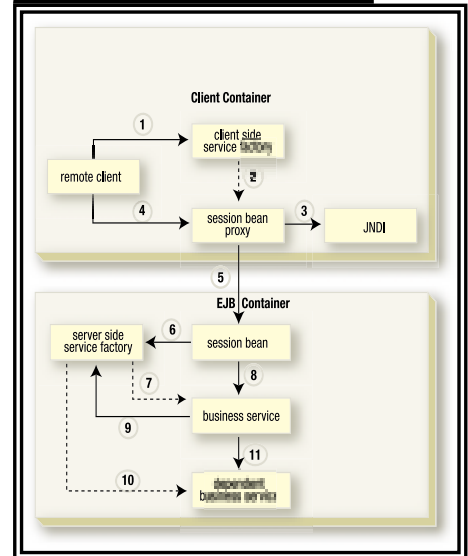
public int guess(int theGuess) throws
    TechnicalException {
    try {
        return this.remote.guess( theGuess );
    }
    catch (RemoteException e) {
        throw new TechnicalException( e );
    }
}
    
```

We rethrow the remote exception to avoid exposing the underlying technology.

The Service Factory

The service factory is responsible for instantiating the correct implementation of a service based on the context in which it is being requested. There are many possible

FIGURE 7



Runtime scenario

implementations. For our example we use simple property files to store name-value pairs. The name identifies the service and the value holds the name of the class implementing the service (the business object, the mockup implementation, or the session bean proxy). The service factory uses reflection to instantiate the proper implementation (see Listing 3).

Every J2EE container has its own instance of service factory with a different configuration (defined by the property files).

Putting It Together

The framework can be seen as a layered architecture. The real communication is between the client and the business implementation. To add J2EE enterprise services (such as transactions, security, etc.), we may instantiate a session bean proxy/session bean layer. If the client and the business logic live in two different containers, we may use remote proxies (remote object + ejb object) provided by J2EE.

The collaboration diagram in Figure 7 illustrates a runtime scenario. In this scenario, a client running in a remote container uses a business service implemented in an EJB container. The business service uses another service (dependent business service) running in the same container.

In Figure 7:

1. The client requests a service from the client-side service factory.
2. The service factory creates the session bean proxy.
3. The service factory uses JNDI to look up the home and create the remote object.

Informatica

www.informatica.com

4. The client invokes a business method on the session bean proxy.
5. The session bean proxy uses the remote object for the session bean obtained in step 3 to forward the business request.
6. The session bean uses the server-side factory to create the business service.
7. The server-side factory creates the business service and returns it to the session bean.
8. The session bean forwards the business request to the business service.
9. The business service needs another service to fulfill the business request. It asks the server-side factory for the dependent service.
10. The server-side factory creates the requested service (no session bean is required).
11. The business service calls business services on its dependent service.

In our simple example, the number guessing game does not need help from other services, so steps 9 through 11 are not required.

Generative Programming

We seem to have created many artifacts to implement a simple service. The effort can be greatly minimized by use of generative programming. We can write a program to generate the session bean (home interface, remote interface, and the session bean implementation; XML fragments for the deployment descriptor) and the session bean proxy. Here are a few approaches that we have had success with:

- Create a simple program that uses reflection upon the business logic bean or the service interface
- Write a script in our favorite case tool
- Write an explicit tool to generate the J2EE connectors (as in OMG's Model Driven Architecture)

Conclusion

In this article, we have presented an approach to service-oriented architecture that yields the following benefits:

- **Shields developers from the complexity of J2EE:** The J2EE code can be generated.
- **Clients only see the factory and the interface to the service.**
- **Business logic developers use standard JavaBeans to implement the interface.**
- **Offers support for concurrent development:** Service providers can offer mockup implementations for the service early in the project's life cycle that enable dependents to continue their work prior to the completion of the real service implementation.
- **Avoids session bean call overhead on service-to-service calls:** The server-side factory would only issue business logic beans, removing the overhead of calling through a session bean (this advantage is minimal with EJB 2.0's local beans).
- **Minimize impact of post-optimization:** We now have a layer of code running on the client side. We can easily move algorithms and data to the client side (either by adding implementation to the session bean proxy or by instantiating the service on the client side).

A complete implementation of the example can be downloaded from www.sys-con.com/weblogic/sourcec.cfm. 🍷

Listing 1

```
class NumberGuessingGameImpl
implements NumberGuessingGame, Serializable {
    public NumberGuessingGameImpl() {}
    public void initialize( int min, int max ) {
        this.secretNumber = (int)
            ( (Math.random() * (max - min) ) + min );
    }
    public int guess( int aGuess ) {
        if (aGuess < this.secretNumber) {
            return -1;
        }
        if (aGuess > this.secretNumber) {
            return 0;
        }
        return 1;
    }
}
```

Listing 2

```
NumberGuessingGameSBP() throws TechnicalException {
    try {
        InitialContext ctx = new InitialContext();
        Object corbaStub = ctx.lookup("NumberGuessingGame");
        NumberGuessingGameHome home =
            (NumberGuessingGame) PortableRemoteObject.narrow(
                corbaStub,
                NumberGuessingGameHome.class);
        this.remote = home.create();
    }
    catch (Exception e) {
        throw new TechnicalException( e );
    }
}
```

Listing 3

```
synchronized public static Object lookup(String nameOfService ) throws
    TechnicalException {
    try {
        String cName = getSetup().getProperty(nameOfService);
        Class c = Class.forName( cName );
        return c.newInstance();
    }
    catch( Exception e ) {
        throw new TechnicalException(e);
    }
}
```


Quest Software

<http://java.quest.com/jprobe/wldj>

IMPROVED PERFORMANCE WITH THE EJBSELECT DATA AGGREGATION PATTERN

INCREASED PERFORMANCE AND BETTER TRANSACTION MANAGEMENT



BY TOM PURCELL

AUTHOR BIO...

Tom Purcell is an architect with Chariot Solutions, a Java consulting company. His 18-year IT journey has tracked the course of the industry from the glass house of the '80s to the client/server of the '90s and on to the J2EE enterprise applications of today. He has designed and developed applications on numerous platforms and worked with Java technology extensively since he installed the first IBM mainframe JVM in 1997.

CONTACT...

tpurcell@chariotsolutions.com

When EJB 2.0 arrived I thought, "Wow, EJB has finally arrived!" With Container Managed Relationships (CMR) and a standard query paradigm I thought the nirvana of portable data layer-independent applications was just a few keystrokes away.

Well, time passed, and while CMR made navigating a foreign key relationship just a "get" away, I still found myself laboriously looping through Collections beans. Calling findByPK on every bean; tossing out the ones the application didn't want; and running up the in cache count for a lot of linking beans that I did not need anyway. How is this better than SQL? I want my JDBC!

Then one night, later than I would have liked, in an effort to recover from a TimedOutException without resurrecting my timeworn JDBC framework again, I stumbled upon something that brightened my world and restored my faith in humanity: the EJSelect.

Intent: Maintain CMP Transaction Support While Boosting Performance

The thing I've always loved about EJB Container Managed Persistence (CMP) has been that it handles all that nasty transaction stuff. None of that "Oh, I forgot to roll that part back" stuff. Of course, the painful part is that some-

times we're forced into some unwieldy code, with lots of nested loops, to get at the data we need. This leads to bulky code, which inevitably leads to bulky performance.

The bulky performance comes from the fact that in all that looping, you're loading lots of stuff just so you can navigate to other stuff. You don't need or care about it; you just need to navigate the schema so you can get what you really want. I don't know about you but Tom's first rule of performance tuning is "Don't make the machine do something it doesn't have to." Doing joins in application code blows that tenet to a place where the donuts are stale and the coffee is cold.

Applicability: Tempted to Resort to JDBC

This "Let's just use JDBC" thing bothers me. I'm not bashing JDBC. I love JDBC. It can be the right tool for the right situation and EJB can be like using a sledgehammer on a thumbtack. The problem is, however, that as soon as you start to wander down that BMP road you're left with two options.

First, build your own persistence framework. Been there. Done that. The customer I did it for owns the rights and my current customer doesn't want to hear about the person months that will add to the schedule (read cost).

The second option is to wing it and hope that everyone on the team knows what they're doing. Well, I hate to admit it but I've been there and done that too and I'm not going back.

Structure

In Figure 1, for brevity's sake I distilled the various EJB interfaces into one class for each entity EJB. There are two important points to take away from this picture. First, notice the CMR relationships. Things are chained together nicely in a properly normalized way. Navigation is possible via what used to be called foreign keys. There

is nothing terribly new here.

Second, notice the "ejbSelectCorporationsOpenInvoices". This lives in the CorporationBean class. It is not, and may not be according to the EJB 2.0 spec, visible from the local or remote interface of the bean. To access it you need to implement a business method in the bean class and expose that to the public interface. That is

where "getCorporationsOpenInvoices" comes in. It is simply a pass-through method that calls "ejbSelectCorporationsOpenInvoices", passing the result of the instances "getCorporation" method as an argument.

Consequences: Who Knows What and Is It Fast Enough?

In the listings shown here (the source code for this article is online at www.syscon.com/weblogic/sourcec.cfm) the eventual solution has the application asking a Corporation entity for its "open" Invoices. We could argue over beers about whether a Corporation object is the right place to put the "is the invoice OPEN?" question, but the point here is I needed it to perform. The Invoice is a more philosophically pure place to encapsulate that question. That being said, this way rocked.

But was it fast enough? Okay, this is where I pull out my standard Golden Hammer Response. This is not a silver bullet. There are cases where resorting to JDBC may be your best option. In this case, however, this way rocked.

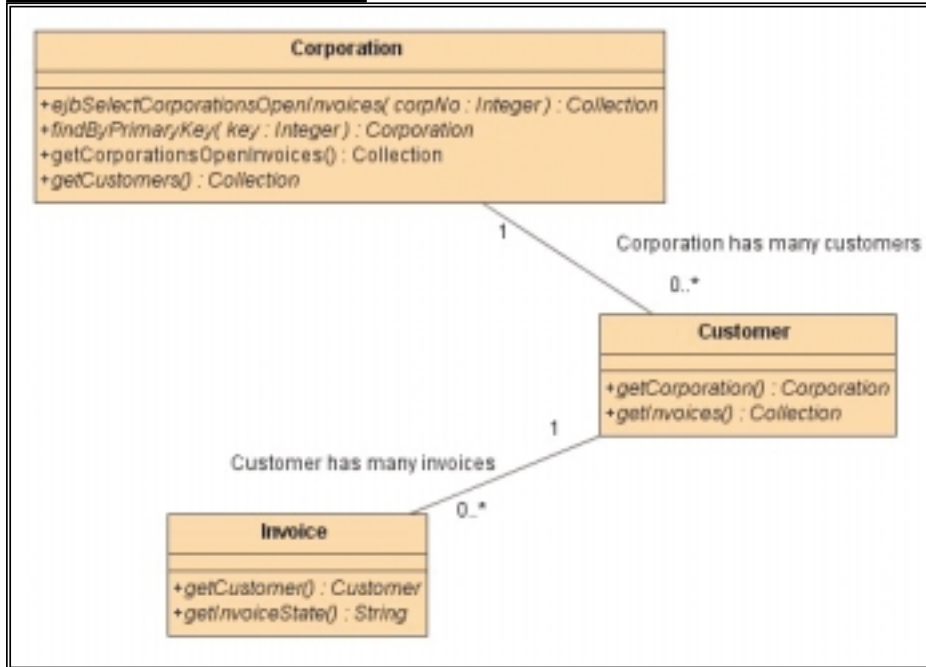
Implementation

Okay, enough of me waxing eloquent, it's time to roll up our sleeves and dig into some code. First a little background. The customer for whom I was staying up late is in the food distribution business. They supply all the burgers, fries, and other good stuff to some of the largest restaurant chains in the country. The food distributor must deliver food to the individual restaurants but receives payment from the restaurant chains' corporate entity. So Invoices are issued at the Customer level and payments are received at the Corporation level. (Note that in the real world there is a hierarchy of Corporations, Franchise Groups, Divisions, and Customers. Payment and invoicing levels vary from chain to chain so we can't just slap corporation on the Invoice table.) When a payment comes in we need all the OPEN Invoices for the Corporation.

Another thing you want to know, since I'm going on about performance, is what kind of box the tests were run on. It's a Dell Inspiron 8200 with a 1.7GHz Pentium 4 and 512MB, of RAM. It's running Windows XP Pro, WebLogic 7.0 SP2, and hitting a local SQLServer 7.0 database. For the test I have one Corporation with 500 customers, each with five invoices (four CLOSED, one OPEN) for a total of 2500 invoices.

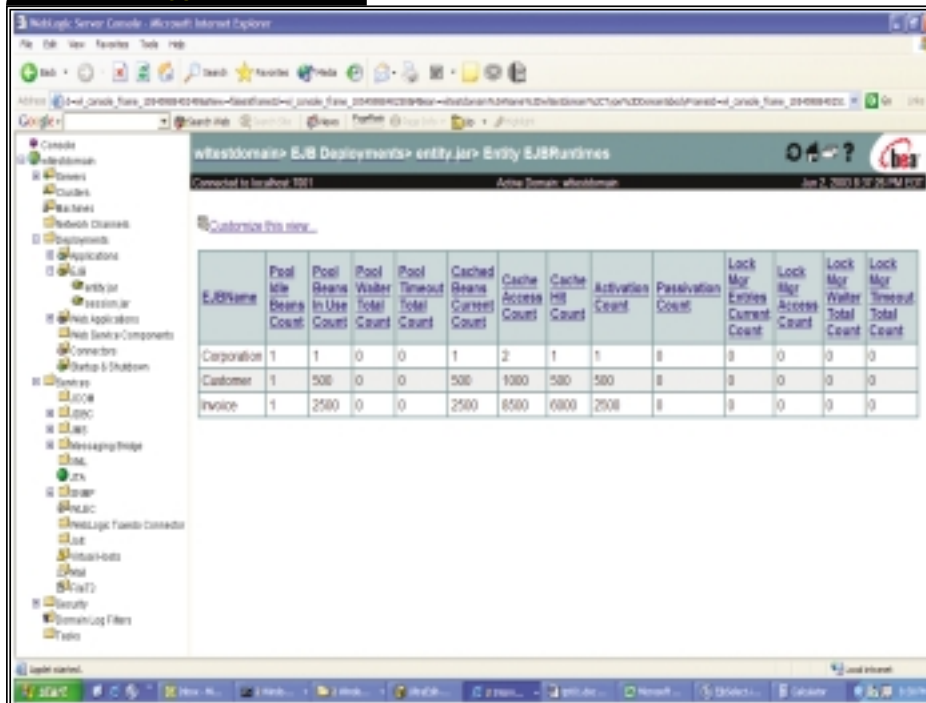
The first approach is a bit EJB 1.1-ish (see Listing 1). In our session bean code we

FIGURE 1



EJB interfaces

FIGURE 2



Rewritten with EJB Select

FIGURE 3

	Count	Count	Count	Count	Count	Count	Count	Count	Count	Count	Count	Count	Count	Count	Count
Customer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Invoice	1	500	0	0	500	400	200	500	0	0	0	0	0	0	0
Corporation	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0

Doing the join in the code

start out by getting a reference to the Corporation. We then use Corporation's CMR relationship with Customer to get a Collection customer linked to the corporation. That's EJB 2.0 but then we loop through the customers. For each customer we get all of their invoices. Then we check each invoice to see if it's opened, keep the ones that are, toss out the ones that aren't.

EJB1.1 Joins are in the application code. Why can't we just use JDBC? The performance was poor too. On my test machine it took an average of 4,126 milliseconds to complete the operation and no wonder. Look at Figure 2. We loaded 500 customers just to get their invoices. Then we sorted through a pile of 2,500 invoices to find the 500 we wanted. Ugh!

Now let's look at a different approach. Remember the old days when you could just write a SQL to do something like this? Well, everything old is new again. EJBSelect is here and you can do that join in a deployment descriptor. Let's look at more code.

In Listing 2 you'll find the engine behind the EJB Select. EJBQL allows us to do joins by navigating the Abstract Schema the way we used to navigate the database schema. What it is doing is selecting the invoices with a state of "O"PEN that are "IN" the result set that is returned by joining Corporation, Customer, and Invoices on their foreign keys. Kind of like doing a JDBC call from your code.

Now look at Listing 3 and see what it does to our session bean code. We now have one loop and no if statements. We go from nine lines of code down to five. I've been writing applications since two character dates were a good idea and it's always been true: the more lines of code you can delete, the better.

"Neat trick, but what about performance?" Glad you asked. As a result we get just what we need. Look at Figure 3. The query loaded 1 Corporation bean and exactly 500 Invoice beans. The operation took an average of 337 milliseconds. That's 1,225% faster! I say that rocks.

Conclusion: Look Before You Jump

Anytime anyone makes a performance claim, even someone as honest and hard working as your humble author, everyone says "Oh, yeah. How did you cook those numbers?" Well I didn't and it's

running in production and the nasty `TimedOutException` went away and I can go to sleep at a reasonable hour... Sorry, but even a salesperson couldn't cook the numbers by 1,225%. Well, okay maybe, but it would be tough.

Once again, I'm not saying it's a silver bullet or a golden hammer but it sure is a pretty nifty pair of pliers. Next time you're tempted to do the JDBC thing give it a shot. It may be a way to get that performance without losing CMP transaction handling. 🍌

Listing 1

```
Corporation corp = getCorporation(corpNo);
Collection custs = corp.getCustomers();

for(Iterator cIt = custs.iterator();cIt.hasNext();) {

    Customer cust = (Customer) cIt.next();
    Collection cInvs = cust.getInvoices();

    for(Iterator iIt = cInvs.iterator();iIt.hasNext();) {
        Invoice inv = (Invoice) iIt.next();

        if (inv.getInvoiceState().equals("O")){
            invoices.add(inv.getAll());
        }
    }
}
```

Listing 2

```
<query>
<query-method>
<method-name>
    ejbSelectCorporationsOpenInvoices
</method-name>
<method-params>
<method-param>
    java.lang.Integer
</method-param>
</method-params>
</query-method>
<result-type-mapping>
    Local
</result-type-mapping>
<ejb-ql>
    select object(inv)
    from Invoice as inv,
        Corporation as corp,
    IN (corp.customers.invoices) as corpsInvs
    where corp.corporationNo = ?1 and
        inv MEMBER OF corpsInvs and
        inv.invoiceState = 'O'
</ejb-ql>
</query>
```

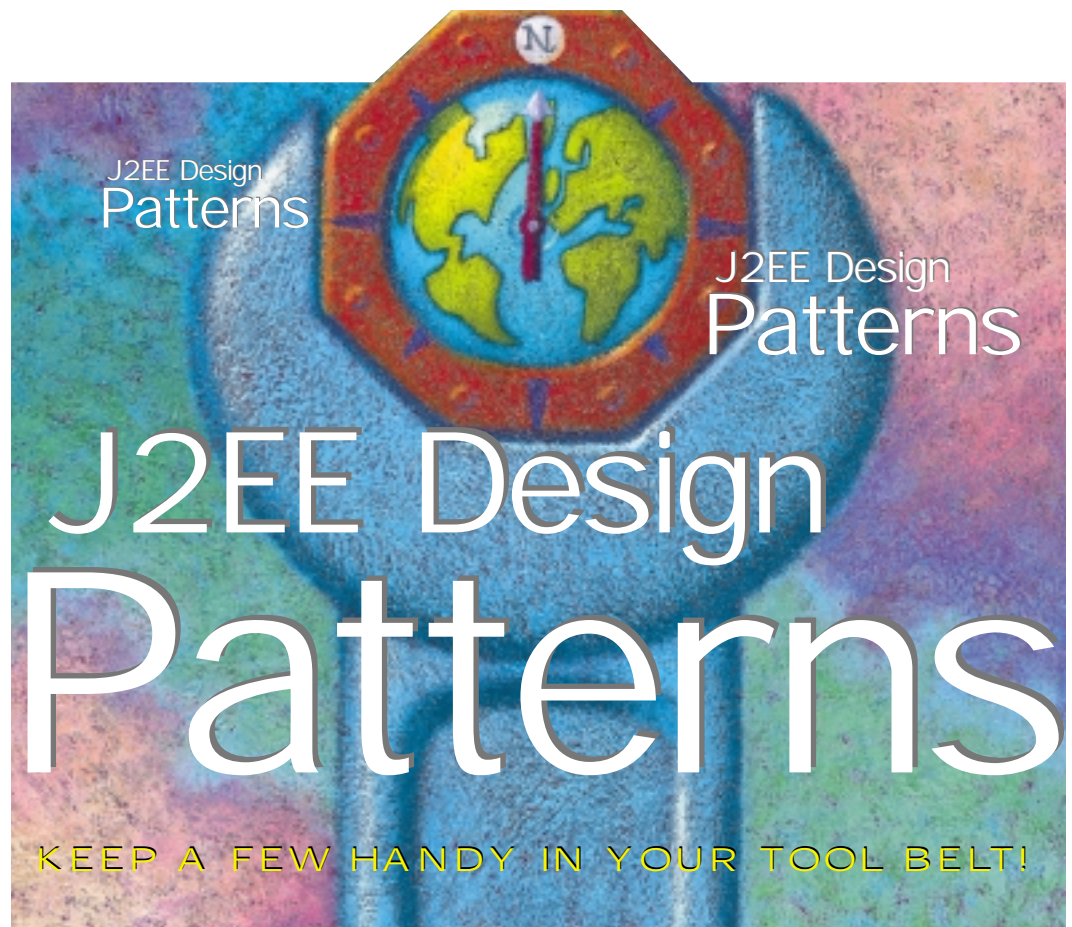
Listing 3

```
Corporation corp = getCorporation(corporationNo);
Collection openInvoices = corp.getCorporationsOpenInvoices();

for(Iterator custOpenInvIt = openInvoices.iterator();
custOpenInvIt.hasNext();) {
    Invoice inv = (Invoice) custOpenInvIt.next();
    invoices.add(inv.getAll());
}
```


Intel

www.intel.com/ad/bea



BY ALAN BAUMGARTEN

AUTHOR BIO

Alan Baumgarten has over 20 years' experience in software engineering and technical training. He currently teaches and develops courseware for /training/etc, Inc., a Columbia, MD-based firm specializing in software training for the enterprise. Alan holds a masters degree in computer science from The Johns Hopkins University.

CONTACT...

alan@trainingetc.com

Not surprisingly, the concept of design patterns originated in the world of architecture. A design pattern is a proven, reusable solution to a recurring problem.

I've always enjoyed building things, both as a software engineer and as a homeowner. Whether the raw materials are Java classes or plywood and nails, I get the best finished results when I start with a good design and follow standard procedures that reflect the experience and knowledge of seasoned professionals. When I wear my tool belt, I try to follow these "tricks of the trade." As a software developer, I use design patterns.

Software design patterns were initially popularized by the book *Design Patterns: Elements of Reusable Object-Oriented Software* by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. The four authors are often referred to as the "Gang of Four," or simply GOF.

Sun's Java Center has cataloged a number of design patterns specifically for J2EE applications (see <http://java.sun.com/blueprints/corej2eepat-terns>). Each pattern is documented formally with accompanying UML diagrams. I'll introduce four of these patterns in this article.

Pattern #1: Composite View Pattern

Don't start from scratch. If you want to construct a housing development with a number of similar houses, you probably wouldn't undertake each house as a separate project. Instead, you would work from a master design and reuse common elements such as doors, windows, and floor plans. That's the idea behind this pattern.

Today's Web applications typically have complex screens with repeated elements such as menus, navigation bars, company logos, and so on. The Composite View pattern suggests that you divide the screens into pieces, coding the pieces as self-contained units. This pattern helps your Web application's screen to have a common look and feel, and it promotes reuse of view components.

It's easy to implement the Composite View pattern using JSPs. There are two built-in mechanisms: the `<%@ include >` directive and the `<jsp:include >` action. Both allow you to construct a JSP from a number of smaller JSPs (see Listing 1). Note that there is a subtle difference between JSP directives and JSP actions. Directives are acted upon when the JSP is translated to a servlet. This normally occurs just once during the lifetime of a Web application. JSP actions, however, are carried out at request time. Request-time inclusion gives you more flexibility and is generally the preferred approach.

A more sophisticated implementation of the Composite View pattern allows you to generate all of an application's screens dynamically, based on settings in an XML-based configuration file. One JSP can be used as a template that specifies the common overall layout and defines subregions that are filled in at run time. Some of these regions contain common screen elements, while other regions show content that varies according to the user's current activity. You will find this approach in Sun's demonstration J2EE application, the Java Pet Store (see <http://java.sun.com/blueprints/patterns/CompositeView.html>).

Custom JSP tags are used to implement the Composite View pattern in the Pet Store application. Custom tags allow you to extend the JSP language and create your own tags. Each tag is associated with a Java class called the tag handler. Sun provides the JSP Standard Tag Library (JSTL), and BEA provides a set of custom tags that are packaged in a file called `weblogic-tags.jar`. This JAR file contains the tag handler classes and a tag library descriptor (TLD). To use these tags, copy this JAR file to the Web application that contains your JSPs and reference the tag library in the JSPs.

Pattern #2: Intercepting Filter Pattern

Tools that can be adapted with “snap-on” accessories are the most useful and flexible. If you've been in a hardware store lately, you've probably seen sets of tools that share a common battery pack. At one moment it's a drill, but with a quick twist of the wrist you can take off the drill head and turn it into a saw or a sander. This makes me think of the Intercepting Filter pattern, which involves creating pluggable filters that can intercept incoming HTTP requests and outgoing responses. You can add and remove these filters unobtrusively without any changes to existing code.

Servlet filters, introduced with servlet specification version 2.3, are perfect for this pattern. A servlet filter is a small class that allows you to manipulate the request and/or response. A filter object is instanti-

ated and managed by the servlet container, and has a life cycle similar to a servlet. Common uses for filters include logging HTTP requests, validating request parameters, and authorizing or denying access to resources. A simple logging filter is shown in Listing 2. Filters can also be used to modify the response, changing its format and/or contents.

Filters are applied selectively to one or more servlets or JSPs by settings in `web.xml` (see Listing 3). Filters and filter mappings can also be configured in the WebLogic Administration Console by selecting your Web application and then clicking on the link labeled “Edit Web Application Deployment Descriptors.”

Servlet filters provide a mechanism to encapsulate recurring tasks in reusable units, a goal shared by the Composite View and several other J2EE design patterns. Since filters are applied declaratively rather than programmatically, they can easily be enabled, disabled, or replaced, just like the attachments on my cordless drill.

Pattern #3: Session Facade Pattern

If you want to have a custom home built, you usually hire a general contractor, who then worries about dealing with all the various subcontractors (drywall, roofing, plumbing, and so on). This simplifies your life in that you only have to interface with and pay the general contractor. The general contractor serves as a front end, or “facade”, for the subcontractors.

A commonly used J2EE pattern for EJBs is called the Session Facade pattern. In this pattern, a stateless session bean is used as a facade to hide access to one or more entity beans. Remote clients interact only with the session bean and never directly with the entity beans (see Figure 1). The first important benefit of the Session Facade pattern is that the client has a simpler interface and doesn't need to know about the details of which entity beans and which methods to use. In my analogy, the session bean is like the general contractor and the entity beans are the subcontractors.

The Session Facade pattern improves performance as only the calls between the client and the session bean go across the network, while calls from the session bean to the entity beans are local to the EJB container. Performance can be further enhanced through the use of local interfaces, introduced as part of the EJB specification version 2.0. Local interfaces provide support for “lightweight” access from with-

in the EJB container, avoiding the overhead associated with a remote interface.

A third benefit of the Session Facade pattern relates to transactions. With container-managed transactions, the container begins a transaction when a method starts to execute and ends the transaction when the method returns. By enclosing the entity beans calls in the session bean method, the database operations are automatically grouped together as a transactional unit. To ensure that they participate in the same transaction, the entity bean methods should be assigned the “Required” transaction attribute in `ejb-jar.xml` (or by using the WebLogic Administration Console).

Pattern #4: Service Locator Pattern

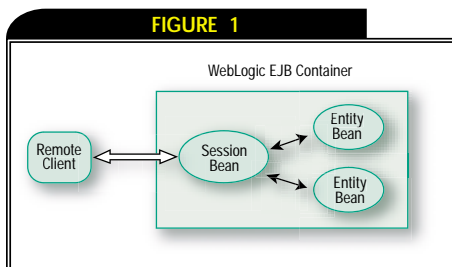
Don't waste a lot of time looking for your tools. In an ideal world, if I needed my extra-long Phillips screwdriver, I would look for it just once and keep it close at hand until I finish my project. Unfortunately, I usually end up spending a lot of time looking for (and cursing at) the same tool. Distributed applications often exhibit the same behavior: they waste time looking for network resources. The Service Locator pattern solves this problem.

Most J2EE applications can take advantage of the Service Locator pattern. The idea behind this pattern is to provide a utility class that centralizes and hides all the details of a complex service. JNDI lookups are perfect candidates, since they generally need to be done only once, and the results can be saved for later use. Enterprise applications use JNDI lookups to locate EJB home interfaces, JDBC data-sources, and Java Message Service (JMS) components.

Listing 4 shows a simple implementation of the Service Locator pattern. Note that the class does not offer a public constructor, but instead has a public static method that allows callers to access a unique instance of the class. This demonstrates a basic GOF pattern, the Singleton pattern.

Conclusion

We've looked at four J2EE design patterns, and I hope you'll be motivated to learn other patterns. I like to keep design patterns handy in my virtual “tool belt,” and I always have room for more. In both software and home construction, design patterns help us to build better products and profit from the experience (and mistakes) of our colleagues. Now if I could just stop hitting my thumb with the hammer! 🍷



Session Facade pattern interaction

Listing 1: Constructing a JSP from a number of smaller JSPs

```
<table>
  <tr>
    <td><jsp:include file="navbar.jsp" %></td>
  <tr>
    <td><jsp:include file="logo.html" %></td>
  <tr>
    <td><jsp:include file="menu.html" %></td>
</table>
```

Listing 2: Logging filter

```
package examples.filters;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoggingFilter implements Filter {

    public void doFilter (ServletRequest request,
        ServletResponse response, FilterChain chain)
        throws ServletException, IOException {

        HttpServletRequest req =
            (HttpServletRequest) request;

        String host = req.getRemoteHost();
        StringBuffer reqURL = req.getRequestURL();

        System.out.println (reqURL + " accessed by "
            + host + " " + new Date());

        chain.doFilter (request, response);
    }

    public void init (FilterConfig config)
        throws ServletException { }

    public void destroy() { }
}
```

Listing 3: Applying a filter in web.xml

```
<filter>
  <filter-name>logger</filter-name>
  <filter-class>
```

```
examples.filters.LoggingFilter
  <filter-class>
</filter>

<filter-mapping>
  <filter-name>logger</filter-name>
  <servlet-name>servlet1</servlet-name>
</filter-mapping>

<filter-mapping>
  <filter-name>logger</filter-name>
  <servlet-name>servlet2</servlet-name>
</filter-mapping>
```

Listing 4: Implementing the ServiceLocator pattern

```
import javax.naming.*;
import javax.sql.*;

public class ServiceCache {

    private static ServiceCache myCache = null;
    private static DataSource ds;

    public static DataSource getDataSource() {
        return getInstance().ds;
    }

    private ServiceCache() { }

    private static ServiceCache getInstance() {

        if (myCache == null) {
            myCache = new ServiceCache();
            myCache.buildCache();
        }
        return myCache;
    }

    private void buildCache() {

        Context ctx = null;

        try {

            ctx = new InitialContext();

            ds = (DataSource) ctx.lookup ("jndiName");

        } catch (NamingException e) {

            // handle exception

        } finally {

            if (ctx != null) {
                try {
                    ctx.close();
                } catch (NamingException ignore) { }
            }
        }
    }
}
```

SAVE 16% OFF**ColdFusion** Developer's Journal**12 Issues for
\$89⁹⁹**

- Exclusive feature articles • Interviews with the hottest names in ColdFusion
- Latest *CFDJ* product reviews • Code examples you can use in your applications • *CFDJ* tips and techniques

That's a savings of **\$17⁸⁹** off the annual newsstand rate. Visit our site at www.sys-con.com/coldfusion/ or call **1-800-303-5282** and subscribe today!

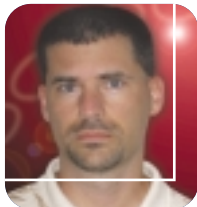
Java Developer's Journal

www.sys-con.com/java



THE PROMISE OF PATTE

AN IDEA THAT SHOULD NOT



BY BILL WILLIS

AUTHOR BIO...

Bill Willis is the director of engineering for ObjectVenture Inc., where he spends a great deal of time developing new pattern-based tools and technology. He is also the director of PatternsCentral, an online community and portal devoted to helping people make more effective use of patterns.

CONTACT...

bwillis@objectventure.com

The concept of patterns has been around as long as we humans have been around. In fact, just about everything we do is centered on recognition, repeatable processes, and routine.

Patterns as a formal concept, however, are a recent development that relatively few professional disciplines have begun to leverage – the most active of which happens to be the software industry. Ever since the now famous building architect Christopher Alexander made an impression on us, numerous books, articles, and training courses have emerged to catalog patterns and educate us in their use.

An interesting thing to note is that although many software developers have heard of patterns and have seen the advertised benefits, most of them still don't fully understand what they are and how to effectively apply them in their daily work. Despite the exciting promise of patterns,

they are still a somewhat elusive concept for many.

What I'll attempt to do in this article is introduce what patterns really are, clear up some common misconceptions, and begin to explore what the software industry needs to do to finally bring the benefits of patterns to the general body of software developers. You may find some of what I have to say a bit surprising.

What a Pattern Is

There are many interpretations of what a pattern is, but one of the better definitions I have seen comes from Alexander himself. The following definition was applied to building architecture but it is equally well suited to software.

Each pattern describes a problem which occurs over and over again in our environment and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

Patterns are becoming popular in software mostly because they encapsulate the experience (years of hard-won experience) of those who have gone before us. This experience provides



proven solutions to common problems many of us face. Among other things, patterns tell us why a given solution works, what its consequences are, and when it is appropriate for use. The value of patterns cannot be underestimated in our craft (or any other) because they really form a collective body of knowledge that no software developer should be without.

An important quality of patterns is that they rarely work alone. You will often use more than one pattern to address different aspects of larger problems. Indeed, properly combining patterns together helps tame complexity and facilitates the communication of a design. Alexander introduced the concept of a pattern language to describe relationships among patterns.

What a Pattern Is Not

Patterns are not code snippets, components, or even a specific solution in a programming language like Java. Although the application of patterns may yield these types of products, patterns themselves describe classes of solutions to common problems. Developers new to patterns may look at a source code example or a UML

diagram and treat it as a representation of the pattern. These examples and visual aids either help you understand the pattern or serve as a specific application of the pattern – not as the actual pattern. A pattern may have a number of slightly varying implementations, so as a developer, you must customize the pattern to fit your own particular situation.

Patterns are not a new methodology. They should not be used to drive your designs and architecture. Instead, they should be used to *improve* your designs and architectures *where appropriate*. You should not begin by wondering how you can make use of a certain set of patterns; rather, you should seek out patterns when you run into an awkward design problem, when you are faced with complexity, when your software is not as scalable or extensible as it needs to be, etc. In other words, use patterns and the experience they offer to guide (not control) you as you design and implement software.

Patterns are not invented or created; they are discovered. It may seem like I'm splitting hairs here, but treating patterns this way will give you a better understanding of what patterns are and how they should be approached. The Rule of Three is a rule of thumb introduced by Ralph Johnson, which basically states that a pattern is not a pattern unless at least three independent applications of its solution have been observed. In this way, patterns are mined from existing software (not invented or created).

Benefits of Patterns

I won't bore you with a top ten list here, but there are three main benefits of patterns that I would like to highlight.

- **A Common Vocabulary:** Any pattern worth its salt will have a good name. This name captures the essence of the pattern, and with just a couple of words allows you to communicate what would otherwise take several sentences (perhaps even a diagram or two) and much hand waving. These names form a collective vocabulary that helps developers communicate better and faster.
- **A Teaching Aid:** As I mentioned earlier, patterns encapsulate the experience of others. Less experienced developers become more productive, and work proceeds more quickly and efficiently with

proven solutions to common problems. An important side effect of using a pattern is that you learn how the solution it describes resolves the forces behind the problem it addresses. Patterns don't just give you some code you can plug in; they guide you in applying a general solution to your specific problem.

- **A Higher Level of Reuse:** As opposed to reusing a component or sections of source code, patterns serve to encapsulate designs and architectures that exist at a higher level and that remain generally independent of any particular programming language. This type of reuse is extremely valuable but difficult to achieve. Patterns provide a concrete means of capturing, layering, and communicating these reusable concepts.

The Problem with Patterns

Patterns today are generally misunderstood and misapplied. They have received great interest from software developers lately thanks mainly to their promised benefits. However, there are several barriers to injecting patterns into the mainstream.

Where to Begin

Most developers new to patterns aren't quite sure where to begin. The amount of information about patterns in software is rapidly increasing (just look at all of the books devoted to the topic), and a not-so-trivial amount of this information is suspect.

From Books to Practice – Difficult

After reading and even understanding a pattern (probably in a book), you still need to take the general solution it offers and specialize it for your particular problem. Remember, patterns provide general solutions (not specific ones) to common problems. I actually treat this as a benefit, but many inexperienced developers have trouble making the jump.

Number of Patterns Increasing

The number of patterns applicable to software is increasing day by day. Some are even redundant, poorly named, or poorly documented. When you add in the fact that patterns are scattered throughout many books, articles, and the Web, finding what you need can be a daunting task.

“Patterns today are generally misunderstood and misapplied”

THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany
collaboration on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks
and more!

Now in More
than 5,000

Bookstores Worldwide –
Subscribe **NOW!**

Go Online & Subscribe **Today!**

*Only \$149 for 1 year (12 issues)
– regular price \$180.



WebLogicDevelopersJournal.com

SYS-CON Media, the world's leading publisher of
IT-technology magazines for developers, software architects,
and e-commerce professionals, brings you the most
comprehensive coverage of WebLogic.



DESIGN PATTERNS

No Support from Tools or Technology

Oddly enough, many product companies are silent on the topic of patterns. Given some of the problems noted above, tools and technology are well suited to assisting software developers in harnessing the experience that patterns offer. They can help you navigate what is available, find what you need, guide you in applying what you have chosen, visualize the consequences, and even help you learn along the way.

Some Recommendations

We need central sources of high quality information on patterns. Places like Hillside.net and PatternsCentral are prom-

ObjectVenture (www.objectventure.com) creates J2EE technology that leverages patterns to deliver on the promise of reusable object technology. ObjectAssembler (OA), the company's flagship product, is a visual, two-way, pattern-based development environment that fully supports the design, development, assembly, and deployment of enterprise and Web-based applications. OA's advanced pattern support allows the creation, addition, enhancement, and enforcement of software patterns. It allows developers to design, construct, and modify applications using visual representations of J2EE patterns, components, and assemblies; thus automating many of the tedious and error-prone implementation tasks.

OA enables organizations to increase development productivity by enforcing design frameworks and models validating component development, flagging errors on a real time basis and automating much of the process for creating and managing component artifacts and assemblies. It provides integrated deployment support for both WebLogic and JBoss, offers complete support for the J2EE component-based architecture, and comes integrated with both Sun's Java Center Core J2EE Pattern Catalog and Apache's Jakarta Struts Framework. OA operates as a stand-alone IDE seamlessly integrated with NetBeans or as a plug-in for either Borland's JBuilder or Sun ONE Studio. With ObjectAssembler, architects, designers, and developers can harness the power of patterns to simplify, accelerate, and enforce their enterprise and Web development efforts. ObjectVenture is the sponsor of the growing patterns community, www.patternscentral.com

ising efforts, but we need to get more of the development community involved in supporting and improving them.

We need centralized (perhaps even distributed) repositories that offer peer-reviewed, well-documented patterns and pattern languages. These repositories should be accessible and should provide a range of tools for navigating them (including visualization techniques).

We need tools and technologies that help us make effective use of patterns. Although the demand for patterns has increased noticeably of late, the support for them in products has lagged considerably. Whether we are creating software from scratch or refactoring existing software, pattern-based tools and technology have the potential to revolutionize the way we create and maintain software.

In Closing

As a WebLogic (and J2EE) developer, you are fortunate to have some excellent resources on patterns. After studying pattern concepts as described by Alexander and some of the fundamental patterns and principles as presented by the Gang of Four and Shalloway, I would highly recommend taking a look at Alur and Marinescu for patterns specific to the J2EE platform. Do yourself a favor and don't miss out on what patterns have to offer.

References

- Alexander, Christopher: www.pattern-language.com/leveltwo/ca.htm
- Alexander, Ishikawa, and Silverstein (1977). *A Pattern Language*. Oxford University Press.
- Johnson, Ralph: <http://st-www.cs.uiuc.edu/users/johnson>
- Hillside.net: <http://hillside.net/patterns>
- PatternsCentral: www.patternscentral.com
- Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Shalloway, Alan; and Trott, James R. (2001). *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Addison-Wesley.
- Alur, Deepak; Crupi, John; Malks, Dan. (2003). *Core J2EE Patterns, Best Practices and Design Strategies*. Second Edition. Prentice Hall.
- Marinescu, Floyd (2002). *EJB Design Patterns: Advanced Patterns, Processes, and Idioms*. John Wiley & Sons. 🍷

Isavix Corporation

www.isavix.net



ADMINISTRATION

Extending the functionality of the WebLogic Server Admin Console provides you with the ability to incorporate the management of your custom application into one central location.

Extending the Admin Console for Your Custom Security Provider

TWO WAYS TO ADD TO YOUR FUNCTIONALITY

BY MIKE KENNEDY



AUTHOR BIO

Mike Kennedy is a senior developer relations engineer for BEA Systems and has over 18 years of software development experience. Mike joined BEA's Professional Services Team in 1999, doing Oracle and J2EE development prior to joining the WebLogic Server Support Team.

CONTACT...

mike.kennedy@bea.com

Even though this example makes use of a TxDataSource, we [BEA] do not support the use of WLS resources by security providers. Any operation which goes through a WLS protected resource, such as a connection pool, is prohibited. You can and should directly access the database via JDBC.

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

We'll look at what is involved in modifying a Custom Security Provider available on dev2dev by adding MBean attributes to manage the TxDataSource, DatabaseUser, DatabaseUserPassword, DatabaseDriver, and DatabaseURL that are currently hard coded into the custom security provider MBean.

There are two ways to extend the console: one is to have the WebLogic Server console attempt to render a details screen based on custom MBean attributes added to the DbSampleAuthenticator.xml file used to create the MBean. The second is to create a Web app containing the class file and JSPs that the console will display when the custom security provider MBean is invoked. I'll discuss both in detail.

Adding Simple Data Types

First, let's look at just adding custom attributes that represent simple data types to the DbSampleAuthenticator XML file and see how to gain access to these values when we want to access the database, both during initialization of WebLogic Server when a datasource is not yet available and afterwards when users are being authenticated when accessing secured resources.

I'll start by adding onto an example found at http://dev2dev.codelibrary/code/sec_rdbms.jsp. The first thing is to add the attributes to the MBean so we can access them through either the command line with `weblogic.Admin`, or JMX code in a `JavaBean` or a `JSP`.

In the `DbSampleAuthenticator.xml` file, add the following writeable `String` attributes:

- DatabaseUser
- DatabaseUserPassword

- DatabaseURL
- DatabaseJDBCdriver
- DataSourceName

```
<MBeanAttribute
  Name           = "DatabaseUser"
  Type           = "java.lang.String"
  Writeable      = "true"
  Default        = "&quot;scott&quot;"
/>
```

Once you have added the attributes, you can run `ant` against the `build.xml` found in the root directory. Many things happen in this `build.xml` and it may be worth your while to examine this file more closely as MBeans are created, jar'd, and copied to the `$WL_HOME/server/lib/mbean-types` directory. If WebLogic Server is already running, it must be restarted for the custom MBeans to be recognized. Once WebLogic Server is running, you should see a screen similar to that in Figure 1, a new Custom Security Authenticator in the console named `DbSampleAuthenticator`. (*Note:* this is WebLogic Server 8.1.) When you select `DbSampleAuthenticator`, you should see a screen similar to that in Figure 2.

After the `DbSampleAuthenticator` has been created and you select the `Details` tab, you should see Figure 3.

If you look at the Java code generated for `DbSampleAuthenticatorMBean`, you'll see the getters and setters for the various attributes you added.

(How exactly do we access these new MBean Attributes? To better understand how the `MBeanMaker` works, refer to `Developing Security Providers for WebLogic Server` (<http://edocs/wls/docs81/dvspisec/design.html#1171038>)).

How Do We Pass This MBean Around?

There is, of course, more than one way to access these new attributes. The first is via a `JMX` call, covered later; and the second is through the `Security Provider's LoginModule`. The first file to examine and change is `DbSampleAuthenticatorProviderImpl.java`, which is called in the WebLogic Server initialization.

```
public void initialize(ProviderMBean providerMBean,
  SecurityServices securityServices)
```

We'll modify the `AppConfigurationEntry` `getConfiguration`, called here by `public`



AppConfigurationEntry.getLoginModuleConfiguration(). A configuration options map is passed to the LoginModule, a perfect place to add our new MBean so the attributes can be accessed later on. This is all we need to do to pass the MBean to the LoginModule's initialize method. Now I'll look at how to access the attributes.

Accessing the MBean's Attributes

The implementation of the LoginModule will be changed to retrieve the ProviderMBean passed in the Map argument. (see <http://edocs.wls/docs81/dvspisec/atn.html#1153042> [Implement the JAAS LoginModule Interface](#)). From there, I'll pass the MBean attributes to the method responsible for authenticating the user (see Listings 1 and 2).

That's it to get the MBean attributes! To see how these are used, look at DbSampleAuthenticatorDatabase.java, where we use the driver, userid, password, and URL to get a database connection when WebLogic Server initializes when JNDI is not set up and the JDBC datasource cannot be accessed.

Extending the Console so the Custom Page is Displayed

Now the fun begins; what do we need to accomplish this feat? Not too much, actually. You'll need a Web app with some default values in the web.xml file and a class that is called to return the JSP page the console will use to render in place of the default details page (see Listing 4). In this example, the JSP must be able to look up the Security MBean via JMX and set the attributes so that the LoginModule can still access them later on. You may ask, do I need a custom Web app for each custom security provider? No, this single Web app with the Java class can handle every security provider and is called each time a user is authenticated; it's up to this class to determine which Security MBean is called and either return null, or there is no JSP page to render; or return the JSP page for the console to display. You could have one Authentication provider with a custom JSP page and another with the default console pages.

That's all we need to do to determine which MBean is called when you click on the custom or any other security provider, this class is called and you can either override the default generated page or return your own.

Now when you click on the DbSampleAuthenticator link, Figure 4 should be displayed.

Creating the JSP

I used the console extension taglib to make the format more consistent with the other default WLS Console pages. I also cheated a little to make this example simple to follow in that I used the same JSP for my form action as I did to render my form. This can definitely be optimized but it will suffice for the purpose of this example.

1. Get the current attribute values so you can display the values on the form
2. Show the form and add a submit button
3. Process the form values and set the MBean attributes

Get the Attributes

In my JSP, I created a small method that I call to populate a HashMap that I use later on to set the default values for my form (see Listing 4; due to space limitations, Listings 4-6 can be found online at www.syscon.com/weblogic/sourcec.cfm). It's

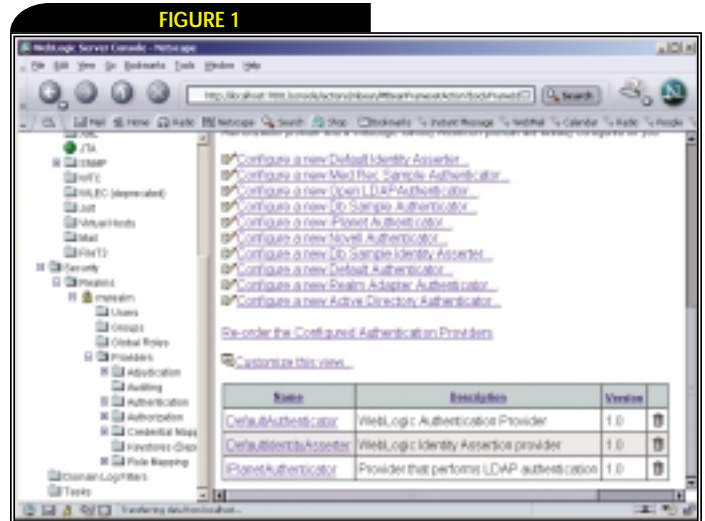


Figure 1
Console showing new DbSampleAuthenticator and DbSampleIdentityAsserter options

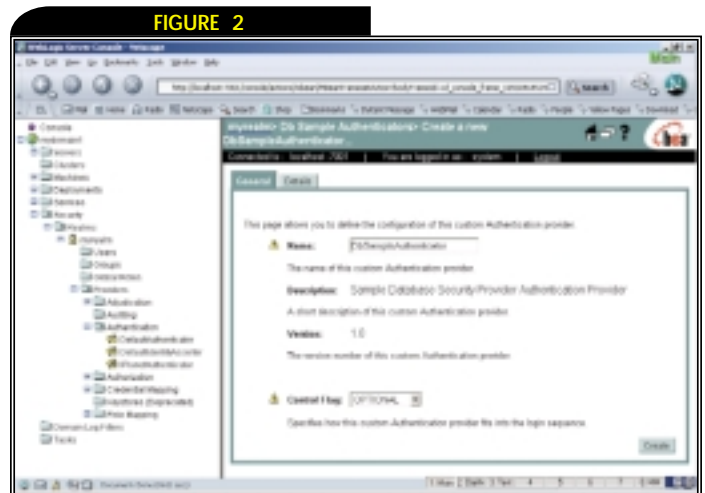


Figure 2
Console showing page once DbSampleAuthenticator link is selected

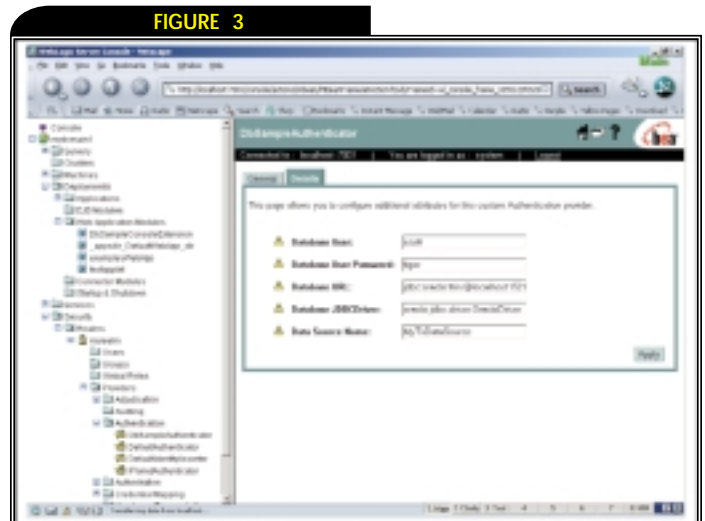
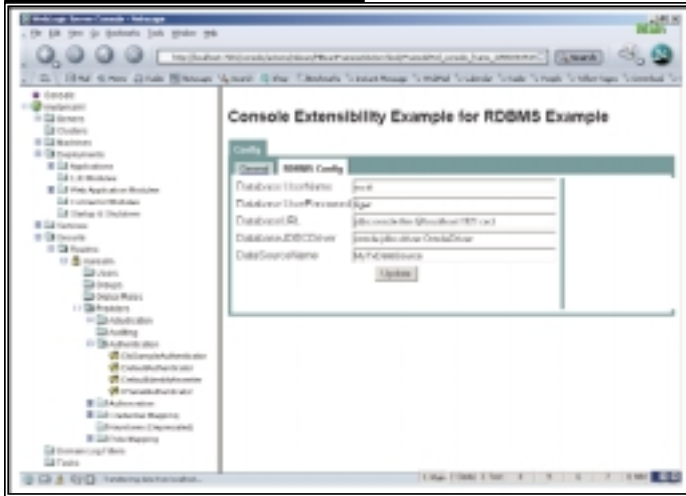


Figure 3
Console showing details page rendered by the console based on the new attributes



FIGURE 4



Console page shown when the user selects the DbSampleAuthenticator link

fairly easy to get the MBean and to retrieve the current attribute values.

Set the Attributes

This was as straightforward as getting the attributes. You must still get the MBean object, but now you create a new attribute

Listing 1: DbSample AuthenticationProviderImpl.java

```
public void initialize(ProviderMBean providermbean, SecurityServices
securityservices)
{
:
myMBean = (DbSampleAuthenticatorMBean)providermbean;
:
}

public AppConfigurableEntry getLoginModuleConfiguration()
{
    HashMap hashmap = new HashMap();
    return getConfiguration(hashmap);
}

private AppConfigurableEntry getConfiguration(HashMap hashmap)
{
    sLogger.debug("Requesting login module configurations");
    hashmap.put("myMBean", myMBean);
    return new
AppConfigurableEntry("examples.db.security.providers.authentication.DbSam
pleLoginModuleImpl", controlFlag, hashmap);
}
```

Listing 2: DbSampleLoginModuleImpl.java

```
public void initialize(Subject subject1, CallbackHandler callbackhan-
dler, Map map, Map map1) {
:
    DbSampleAuthenticatorMBean dbsampleauthenticatormbean =
(DbSampleAuthenticatorMBean)map1.get("myMBean");
:
    String databaseURL = dbsampleauthenticatormbean.getDatabaseURL();
    String databaseUser = dbsampleauthenticatormbean.getDatabaseUser();
    String databaseUserPassword =
dbsampleauthenticatormbean.getDatabaseUserPassword();
    String databaseJDBCdriver =
```

based on the Attribute Name and the new value entered from the form, and then set the attribute. Listing 5 loops through all of the form elements passed in the POST operation and sets those that match the attribute names we want to set.

The last thing to note is that since the JSP instantiates the MBean, you will need to import the MBean class, DbSampleAuthenticatorMBean, which means that the class examples.db.security.providers.authentication.DbSampleAuthenticatorMBean must be in the web-app/WEB-INF/classes directory.

To set the MBeans from the command line, use Listing 6.

Conclusion

In this article I tried to demonstrate how to extend the console for Custom Security Providers by adding new MBean attributes and the code needed to access them. There are two ways to extend the functionality of the console for a custom security provider. You can simply add writeable attributes to the SecurityProvider.xml file and, rebuild the Security Provider jar file and the WebLogic Server Console will attempt to render the screen as a details page. Or, you can write your own Web application with a console extension class and custom JSP files for the console to use.

Acknowledgment

I want to thank Chris Chiodo of the WebLogic Server OA&M team for finding answers to some obtuse questions I had as I developed these examples. 🍌

```
dbsampleauthenticatormbean.getDatabaseJDBCdriver();
    String DataSourceName =
dbsampleauthenticatormbean.getDataSourceName();
    DbSampleAuthenticatorDatabase database = new
DbSampleAuthenticatorDatabase(databaseURL, databaseUser,
databaseUserPassword, databaseJDBCdriver, DataSourceName);
:
}
```

Listing 3: Write the Console Extension class used to return the JSP page the console will render

```
public final class DbSampleConsoleExtension extends Extension
implements SecurityExtensionV2
{
    public DbSampleConsoleExtension()
    {
    }

    public String getExtensionForProvider(ObjectName objectname, String s,
ObjectName objectname1)
    {
        sLogger.debug("DbSampleConsoleExtension.getExtensionForProvider");

        if(s.equals("examples.db.security.providers.authentication.DbSampleAuthenti
catorMBean"))
        {
            return "DbSampleConsolePage.jsp"; // This jsp must be in
this web-app or a 404 will be returned
        } else
        {
            return null; // for all other Providers, have the console
use default behavior
        }
    }
}
```

International Web Services Conference & Expo

Web Services Edge

3rd Annual



Delivering
.NET, JAVA,
Mac OS X,
& XML
Technologies



2003 WEST



SEPT. 30 - OCT. 2, 2003
Santa Clara, CA



REGISTER TODAY!
CALL 201-802-3058
www.sys-con.com
Register before August 1st and
SAVE Up To \$300

Produced and Presented:
SYS-CON EVENTS

Media Sponsors:

WEB SERVICES EDGE CONFERENCE & EXPO

web services
conference & expo

EDGE

SEPT. 30 - OCT. 2, 2003
Santa Clara Convention Center

It is our pleasure to bring the latest edition of the highly successful Web Services Edge Conference & Expo to the Santa Clara Convention Center, September 30 – October 2, 2003. The Third Annual Web Services Edge West Conference & Expo will continue to build on our past success to make available the most current and relevant information to you, our valued attendee.

With the widespread adoption of Web services across the industry, developers are facing new challenges. In this year's conference program, we will address these challenges with our most comprehensive program to date. Web Services Edge 2003 West will provide practical approaches and solutions to overcome the hurdles in developing and deploying Web services in today's competitive markets. Once again Web Services Edge will feature dedicated tracks covering Java, Web Services, .NET, and XML - along with the newly added Mac OS X Track. Sessions in this track will highlight the use of the Mac OS X platform, which combines the ease of use of a Mac with the power of Unix, in applications and Web services development, deployment, and management.

Your three days will include highly informative keynotes, conference sessions, tutorials, industry-leading university certification programs, case studies, and demo presentations. The Expo Hall will be open on September 30 and October 1, featuring the largest grouping of quality exhibitors prepared to field your questions and solve your development needs.

All the best,
SYS-CON Events

FEATURES & ATTRACTIVE

TO ALL CONFERENCE & EXPO REGISTRANTS

- 3 Days Packed with Education and Training
- Keynotes & Panel Discussions from Industry Leaders
- 60 Hard-hitting and Informative Seminars
- FREE .NET Tutorial with Russ' Tool Shed
- Java University Certification Training
- Industry-Leading Certification Programs
- "Birds of a Feather" Discussions
- Round Table Discussions
- Opening Day Welcome Reception
- SAMS Meet the Authors Hot Topics Lounge
- Compelling Case Studies & Best Practices
- Hands-On Labs
- Featured Product Demonstrations
- Exhibit Floor featuring more than 40 companies and hundreds of products
- Real-time SYS-CON Radio Interviews

WHO SHOULD ATTEND

- Software Developer
- Software Engineer
- Development Manager
- Application Developer
- Technical Director
- Analyst/Programmer
- IT Manager
- Technical Architect
- Team Leader
- Software Consultant

HIGHLIGHTED SPEAKERS

Allan Vermeulen

CTO, Amazon.com

amazon.com



CTO and vice president at Amazon.com directly oversees the Platform Technologies group. This group is responsible for guiding Amazon.com's technology architecture, including building and acquiring foundational components. Prior to his move to Amazon.com, Vermeulen was CTO and vice president of development at Rogue Wave Software. He holds a PhD in Systems Design Engineering from the University of Waterloo.

John Schmidt

Leader of Systems Integration and Middleware, Best Buy Co.



John Schmidt is the chairman of the Methodology Committee for the EAI Industry Consortium and leader of systems integration and middleware at Best Buy Co., a leading specialty retailer of consumer electronics, personal computers, entertainment software, and appliances.

Dave Chappell

VP, Chief Technology Evangelist, Sonic Software



Dave Chappell is the vice president and chief technology evangelist for Sonic Software. He has more than 18 years of industry experience building software tools and infrastructure for application developers, spanning all aspects of R&D, sales, marketing, and support services. Dave has also been published in numerous technical journals, and is currently writing a series of contributed articles for *Java Developer's Journal*.

Anne Thomas Manes

Research Director, Burton Group



Anne Thomas Manes is a research director at Burton Group, a research, consulting, and advisory firm. Anne leads research for the Application Platform Strategies service. Named one of NetworkWorld's "50 Most Powerful People in Networking" in 2002, and one of Enterprise Systems Journal's "Power 100 IT Leaders" in 2001, Anne is a renowned technologist in the Web services space. Anne participates in standards development at W3C and OASIS. She is a frequent speaker at trade shows and author of numerous articles and the book *Web Services: A Manager's Guide*.

Register Online at

www.sys-con.com

WEB SERVICES EDGE CONFERENCE & EXPO

web services **EDGE**
conference & expo

SEPT. 30 - OCT. 2, 2003
Santa Clara Convention Center

WEB SERVICES TECHNOLOGY

Presentations will include discussions of security, interoperability, the role of UDDI, progress of the standards-making bodies, SOAP, and BPM. Case studies cover the design and deployment of Web services in the marketplace.

Sessions will focus on:

- Interoperability
- Enterprise Networks
- Web Services Management
- Web Services Standards
- Web Services Orchestration
- Security (WS-Security, SAML)
- BPEL4WS
- UDDI: Dead or Alive?
- ebXML & Web Services
- EAI & Web Services
- RPC vs Documents: Uses and Differences
- User Interfaces for Web Services
- Web Services Best Practices
- Service Oriented Architecture



XML TECHNOLOGY

Presentations will focus on the various facets of XML technologies as they are applied to solving business computing problems. Sessions will include emerging standards in XML Schemas, XML repositories, industry applications of XML, applying XML for building Web services applications, XML/XSLT/XQuery-based programming using Java/.NET, XML databases, XML tools and servers, XML-based messaging, and the issues related to applying XML in B2B/EAI applications. The XML Track is geared for audiences ranging from beginners to system architects and advanced developers.

Sessions will focus on:

- XML Standards & Vocabularies
- Introduction to XForms
- Securing Your XML and Web Services Infrastructure
- XQuery Fundamentals: Key Ingredient to Enterprise Information Integration
- XML and Enterprise Architecture: Technology Trends
- Standards-Based Enterprise Middleware Using XML/Web Services
- XML and Financial Services
- Canonical Documents for Your Business: Design Strategies
- XPath/XSLT 2.0: What's New?
- XML Schema Best Practices
- XML in EAI, Enterprise Portals, Content Management

XML

MAC OS X

OS X represents a new wave of operating systems. It combines the ease of use of a Mac with the power of Unix. Sessions in this track will highlight the use of the Mac OS X platform in applications and Web services development, deployment and management.

Sessions will focus on:

- Introducing OS X (Panther): What's New?
- Quick Applications using AppleScript
- Enterprise Java and OS X
- Developing Web Services Using WebObjects
- Xserve: Ease of OS X and Power of Unix
- Introducing Quartz: 2D Graphics for Apple
- OS X for the Unix Developer
- Securing OS X Applications
- Java and OS X: A Perfect Marriage
- Programming Rich User Interfaces Using Cocoa



JAVA TECHNOLOGY

The Java Track features presentations aimed at the beginner, as well as the seasoned Java developer. Sessions will explore the whole spectrum of Java, focusing on J2EE, application architecture, EJB, and J2ME. In addition the track will cover the latest in SWT, Ant, JUnit, open source frameworks, as well as an in-depth look into the vital role that Java is playing in building and deploying Web services.

Sessions will focus on:

- Enterprise Java 1.4
- Ant Applied in "Real World" Web Services
- Developing Application Frameworks w/SWT
- Empowering Java and RSS for Blogging
- JUnit: Testing Your Java with JUnit
- JDK1.5: The Tiger
- Simplifying J2EE Applications
- Using IBM's Emerging Technologies Toolkit (ETTK)
- Apache Axis
- Meeting the Challenges of J2ME Development
- Integrating Java + .NET
- Squeezing Java



.NET TECHNOLOGY

Presentations will explore the Microsoft .NET platform for Web services. To the average developer, it represents an entirely new approach to creating software for the Microsoft platform. What's more, .NET development products - such as Visual Studio .NET - now bring the power of drag-and-drop, GUI-based programming to such diverse platforms as the Web and mobile devices.

Sessions will focus on:

- ASP.NET
- Security
- VB.NET
- .NET and XML
- Smart Device Extensions for VS.NET
- Best Practices
- Shared Source CLI
- .NET Remoting
- Smart Devices in Health Care Settings
- Mobile Internet Toolkit
- ROTOR
- Portable .NET
- ASP.NET Using Mono
- Using WSE with IBM's WSTK
- GUI applications Using Mono
- Portals - Windows SharePoint Services/SharePoint Portal Server
- Windows Server 2003 and IIS 6
- .NET and Java Interoperability
- Distributed .NET for Financial Applications
- Developing C# with Eclipse

Microsoft
.net



For more information visit
www.sys-con.com
or call
201 802-3069

**SPECIAL
DISCOUNTS
AVAILABLE**

Take advantage of the Early Bird and Pre-registration values available right now, or save even more with a group of 5 or more. For special group discounts contact Michael Lynch at mike@sys-con.com, or by phone at (201) 802-3058.

Conference at-a-Glance

TUESDAY, SEPTEMBER 30 DAY 1

	JAVA	.NET	WEB SERVICES	XML
8:00AM – 4:00PM	REGISTRATION			
9:00AM – 9:50AM	Enterprise Java 1.4	Using WSE 2.0	Web Services Management	Introduction to Xforms
10:00AM – 10:50AM	Opening Keynote - Allen Vermeulen, CTO, Amazon.com			
11:00AM – 6:00PM	EXPO OPEN			
2:00PM – 2:50PM	Keynote Panel Discussion - Enterprise Application Integration			
3:00PM – 3:50PM	Ant Applied in "Real World" Web Services	Smart Devices in Health Care Settings	Service Oriented Architecture	Securing Your XML and Web Services Infrastructure
4:00PM – 4:50PM	Developing Application Frameworks with SWT	Using the Mobile Internet Toolkit	Web Services Orchestration	XQuery Fundamentals: Key Ingredient to Enterprise Information Integration
5:00PM	OPENING NIGHT RECEPTION			

WEDNESDAY, OCTOBER 1 DAY 2

8:00AM – 4:00PM	REGISTRATION			
9:00AM – 9:50AM	Integrating Java and .NET	Introduction to ROTOR	Security (WS-Security, SAML)	Standards-Based Enterprise Middleware Using XML/Web Services
10:00AM – 10:50AM	Morning Keynote			
11:00AM – 4:00PM	EXPO OPEN			
2:00PM – 2:50PM	Keynote Panel Discussion - Interoperability: Is Web Services Delivering?			
3:00PM – 3:50PM	JUnit: Testing Your Java with JUnit	Using Portable .NET	WS-BPEL	XML and Enterprise Architecture: Technology Trends
4:00PM – 4:50PM	JDK1.5: The Tiger	ASP.NET with Mono	UDDI: Dead or Alive?	Using XML Schemas Effectively in WSDL Design
5:00PM – 6:00PM	Squeezing Java	Using WSE with IBM's WSTK	Web Services Choreography, Management, and Security - Can They Dance Together?	Canonical Documents for Your Business: Design Strategies

THURSDAY, OCTOBER 2 DAY 3

8:00AM – 4:00PM	REGISTRATION			
9:00AM – 9:50AM	Using IBM's Emerging Technologies Toolkit (ETTK)	Distributed .NET for Financial Applications	eAI & Web Services	XML and the Fortune 500
10:00AM – 10:50AM	Morning Technical Keynote			
11:00AM – 11:50AM	Apache Axis	Developing C# with Eclipse	RPC vs Documents: Uses and Differences	XPath/XSLT 2.0: What's New?
12:00PM	BREAK			
1:00PM – 1:50PM	Meeting the Challenges of J2ME Development	Windows SharePoint Services	The Seven Habits of Highly Effective Enterprise Service Buses (ESBs)	ebXML & Web Services
2:00PM – 2:50PM	Keynote Panel Discussion - Summit on Web Services Standards			
3:00PM – 3:50PM	Empowering Java and RSS for Blogging	BizTalk 2003	See www.sys-con.com for more information	See www.sys-con.com for more information
4:00PM – 5:00PM	See www.sys-con.com for more information	See www.sys-con.com for more information	See www.sys-con.com for more information	See www.sys-con.com for more information

MAC OS X

Introducing OS X
(Panther) What's New?

Programming Rich User
Interfaces Using Cocoa

Quick Applications
using AppleScript

Java and OS X: A Perfect Marriage

Enterprise Java and OS X

Developing Web Services Using WebObjects

Cocoa, Carbon, Java: Application Frameworks
for OS X (When to use what)

Securing OS X Applications

Xserve: Ease of OS X
and Power of Unix

OS X for the Unix Developer

Introducing Quartz: 2D
Graphics for Apple

See www.sys-con.com for
more information



RUSS' TOOLSHED

TINKERING WITH VISUAL STUDIO

FREE
Microsoft®
Tutorial

RUSS' TOOL SHED

Join Russ as he shows you how to use Visual Studio .NET

INTRO TO WEB SERVICES USING VS.NET

One of the key ideas behind the .NET strategy is the concept of software as a service, or in short, Web services. This session will explain what a Web service is and provide an overview of its related technologies like XML, SOAP, and UDDI. We will demonstrate how the .NET Framework makes it easy to implement them for new and existing applications. This session will also provide concrete best practices for building XML Web services using Visual Studio .NET. We'll answer many common questions like: How will my Web service scale? How can my XML Web services enable interoperability with Web services from other vendors as well as within my own organization? We'll delve into building highly reliable and secure Web services. Also, we will discuss issues such as dealing with complex data types using WSDL (Web Services Description Language), as well as securing SOAP messages using encryption. We'll see how developers can use enterprise-level XML Web services to simplify customer solutions.



ADVANCED WEB SERVICES USING ASP.NET

This session will explore some of the more advanced areas of SOAP in ASP.NET's support for Web services. ASP.NET Web services are the preferred way for Web developers to expose Web services on the Internet. The goal is quick, easy, and high-performing SOAP services. We will look at how to use the SOAP extension classes to create some very interesting applications on top of the core SOAP architecture found within the .NET Framework. For instance, you can implement an encryption algorithm or screen scraping on top of the Web service call. We'll dig into more advanced topics, explore the SOAP headers, and see ways to ensure security in our Web services.

.NET REMOTING ESSENTIALS

Microsoft .NET Remoting is the .NET technology that allows you to easily and quickly build distributed applications. All of the application components can be on one computer or they can be on multiple computers around the world. .NET Remoting allows client applications to use objects in other processes on the same computer or on any other computer to which it can connect over its network. During this presentation we will discuss what you will need to know to get started with .NET Remoting. We will talk about how .NET Remoting compares with DCOM, how to host remoted objects in a variety of applications, how to call remoted objects from a client application, how to control the lifetime of remoted objects, and how to secure remoting applications.

Register Online at
www.sys-con.com

CONFERENCE: Sept. 30 – Oct. 2, 2003 EXPO: Sept. 30 – Oct. 1, 2003

Santa Clara Convention Center • Santa Clara, CA

THREE WAYS TO REGISTER FOR CONFERENCE

- 1) **On the Web:** Credit Cards or "Bill Me." Please make checks payable to SYS-CON Events.
- 2) **By Fax:** Credit Cards or "Bill Me" 201-782-9651
- 3) **By Mail:** 135 Chestnut Ridge Road, Montvale, New Jersey 07645, Attention: Registration

Please note: Registrations are not confirmed until payment is received.

Please complete sections 1, 2, 3 and 4

1 YOUR INFORMATION (Please Print) Mr. Ms.

First Name _____ Last Name _____

Title _____

Company _____

Street _____

Mail Stop _____

City _____

State _____ Zip _____ Country _____

Phone _____

Fax _____ E-Mail _____

2 PAYMENT METHOD: (Payment in full due with registration)

Check or Money Order Enclosed (Registration confirmed upon receipt of payment)

Check # _____ Amount of Check \$ _____

Charge my Visa MasterCard American Express Discover

Name on card _____

Card # _____ Exp. Date _____

Signature _____

Billing Address (if different from mailing address) _____

3 PLEASE INDICATE YOUR CONFERENCE CHOICE

	Before 8/1/03	Before 9/26/03	Onsite
<input type="checkbox"/> GP Gold Passport Good for all three days of the .NET, Web Services, XML, Java, and Mac OS X Tracks, including Keynotes, Panel Discussions, preferred seating for Microsoft .NET's Russ' "Tool Shed" Tutorial, and your choice of one Sun Microsystems Java™ University Class	\$1,195.00	\$1,395.00	\$1,495.00
<input type="checkbox"/> 3D Three Day Conference (Does not include Sun Java™ Education)	\$1,095.00	\$1,295.00	\$1,395.00
<input type="checkbox"/> 2D Two Day Conference (Does not include Sun Java™ Education) (select any two days: <input type="checkbox"/> Tue. <input type="checkbox"/> Wed. <input type="checkbox"/> Thurs.)	\$995.00	\$1,195.00	\$1,295.00
<input type="checkbox"/> 1D One Day Conference (Does not include Sun Java™ Education) (select any one day: <input type="checkbox"/> Tue. <input type="checkbox"/> Wed. <input type="checkbox"/> Thurs.)	\$495.00	\$595.00	\$695.00
<input type="checkbox"/> JU1 Sun Java™ University Class Select one: <input type="checkbox"/> Web Services Programming Using Java™ Technology and XML (Sept. 30) <input type="checkbox"/> Java™ Fast Path: Programmer (Oct. 1) <input type="checkbox"/> Java™ Fast Path: Architect (Oct. 2)	\$595.00	\$695.00	\$795.00
<input type="checkbox"/> JU2 Sun Java™ University Class Select two: <input type="checkbox"/> Web Services Programming Using Java™ Technology and XML (Sept. 30) <input type="checkbox"/> Java™ Fast Path: Programmer (Oct. 1) <input type="checkbox"/> Java™ Fast Path: Architect (Oct. 2)	\$1,095.00	\$1,295.00	\$1,395.00
<input type="checkbox"/> JU3 Sun Java™ University Class Select three: <input type="checkbox"/> Web Services Programming Using Java™ Technology and XML (Sept. 30) <input type="checkbox"/> Java™ Fast Path: Programmer (Oct. 1) <input type="checkbox"/> Java™ Fast Path: Architect (Oct. 2)	\$1,195.00	\$1,395.00	\$1,495.00
<input type="checkbox"/> EO Expo Only	FREE	FREE	\$50.00

Total Registration fee \$ _____

4

A. Your Job Title

CTO, CIO, VP, Chief Architect
 Software Development Director/Manager/Evangelist
 IT Director/Manager
 Project Manager/Project Leader/Group Leader
 Software Architect/Systems Analyst
 Application Programmer/Evangelist
 Database Administrator/Programmer
 Software Developer/Systems Integrator/Consultant
 Web Programmer
 CEO/COO/President/Chairman/Owner/Partner
 VP/Director/Manager Marketing, Sales
 VP/Director/Manager of Product Development
 General Division Manager/Department Manager
 Other (please specify) _____

B. Business/Industry

Computer Software Government/Military/Aerospace
 Computer Hardware and Electronics Health Care/Medical
 Insurance/Legal
 Computer Networking & Telecommunications Education
 Utilities
 Internet/Web/E-commerce Architecture/Construction/Real Estate
 Consulting & Systems Integrator
 Financial Services Agriculture
 Manufacturing Nonprofit/Religious
 Wholesale/Retail/Distribution Other (please specify)
 Transportation
 Travel/Hospitality

C. Total Number of Employees at Your Location and Entire Organization (check all that apply):

	Location	Company
10,000 or more	01 <input type="checkbox"/>	01 <input type="checkbox"/>
5,000 - 9,999	02 <input type="checkbox"/>	02 <input type="checkbox"/>
1,000 - 4,999	03 <input type="checkbox"/>	03 <input type="checkbox"/>
500 - 999	04 <input type="checkbox"/>	04 <input type="checkbox"/>
100-499	05 <input type="checkbox"/>	05 <input type="checkbox"/>
100 or less	06 <input type="checkbox"/>	06 <input type="checkbox"/>

D. Please indicate the value of communications and computer products and services that you recommend, buy, specify, or approve over the course of one year:

\$10 million or more \$10,000 - \$99,999
 \$1 million - \$9.9 million Less than \$10,000
 \$500,000 - \$999,999 Don't know
 \$100,000 - \$499,999

E. What is your company's gross annual revenue?

\$10 billion or more \$1 million - \$9.9 million
 \$1 billion - \$9.9 billion Less than \$1 million
 \$100 million - \$999 million Don't know
 \$10 million - \$99.9 million

F. Do you recommend, specify, evaluate, approve or purchase wireless products or services for your organization?

01 Yes 02 No

G. Which of the following products, services, and/or technologies do you currently approve, specify or recommend the purchase of?

Application Servers
 Web Servers
 Server Side Hardware
 Client Side Hardware
 Wireless Device Hardware
 Databases
 Java IDEs
 Class Libraries
 Software Testing Tools
 Web Testing Tools
 Modeling Tools
 Team Development Tools
 Installation Tools
 Frameworks
 Database Access Tools / JDBC Devices
 Application Integration Tools
 Enterprise Development Tool Suites
 Messaging Tools
 Reporting Tools
 Debugging Tools
 Virtual Machines
 Wireless Development Tools
 XML Tools
 Web Services Development Toolkits
 Professional Training Services
 Other [Please Specify] _____

SYS-CON Events, Inc. and SYS-CON Media make no warranties regarding content, speakers, or attendance. The opinions of speakers, exhibitors and sponsors do not reflect the opinion of SYS-CON Events and SYS-CON Media and no endorsement of speakers, exhibitors, companies, products, or sponsors is implied.

Web Services Edge 2003
SEPT. 30 - OCT. 2, 2003
 Santa Clara Convention Center



**WEB SERVICES EDGE
 CONFERENCE & EXPO**

SYSCON EVENTS

CANCELLATIONS, SUBSTITUTIONS, REFUNDS

If you require special assistance covered under the Americans with Disabilities Act, please call 201-802-3058 by September 16, 2003.

Fax written request to SYS-CON Registration 201-782-9651. Requests for refunds received prior to August 29, 2003 will be honored, less a 10% handling charge; requests received after August 29, 2003, and before September 12,

2003, will be honored less a 20% handling charge. No requests for refunds will be honored after September 12, 2003. Requests for substitutions must be made in writing prior to September 26, 2003. No one under 18 is permitted to attend. No warranties are made regarding the content of sessions or materials.

Speakers, sessions, and schedule are subject to change without prior notice.

No solicitation by anyone other than official exhibitors, sponsors or marketing partners is permitted. Such behavior is cause for expulsion without refund.



The **Leading Magazine** for Enterprise and IT Management



LinuxWorld Magazine

There is no escaping the penetration of Linux into the corporate world. Traditional models are being turned on their head as the open-for-everyone Linux bandwagon rolls forward.

Linux is an operating system that is traditionally held in the highest esteem by the hardcore or geek developers of the world. With its roots firmly seeded in the open-source model, Linux is very much born from the "if it's broke, then fix it yourself" attitude.

Major corporations including IBM, Oracle, Sun, and Dell have all committed significant resources and money to ensure their strategy for the future involves Linux. Linux has arrived at the boardroom.

Yet until now, no title has existed that explicitly addresses this new hunger for information from the corporate arena. *LinuxWorld Magazine* is aimed squarely at providing this group with the knowledge and background necessary to make decisions to utilize the Linux operating system.

Look for all the strategic information required to better inform the community on how powerful an alternative Linux can be. *LinuxWorld Magazine* does not feature low-level code snippets but focuses instead on the higher logistical level, providing advice on hardware, to software, through to the recruiting of trained personnel required to successfully deploy a Linux-based solution. Each month presents a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

Regular features

include:

Advice on Linux Infrastructure

Detailed Software Reviews

Migration Advice

Hardware Advice

CEO Guest Editorials

Recruiting/Certification Advice

Latest News That Matters

Case Studies

SAVE 30% OFF!

REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY

\$49⁹⁹

12 ISSUES/YR

*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

SUBSCRIBE TODAY!

WWW.SYS-CON.COM

OR CALL

1-888-303-5282

FOR ADVERTISING INFORMATION:

CALL 201 802.3020 OR

VISIT WWW.SYS-CON.COM



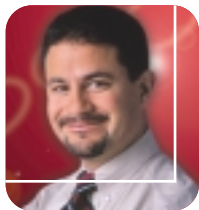
FROM THE OFFICE OF THE CTO

If you were to trace the origins of the excitement about Java, you might go back to May of 1995 when Sun first announced Java at SunWorld '95. The sexiest part of the show was HotJava, a browser written entirely in Java and capable of downloading smart content to the desktop.

The Language, the Server, and the JVM

THERE'S MOVEMENT DEEP WITHIN THE STACK

BY BENJAMIN RENAUD



AUTHOR BIO

Benjamin Renaud is a strategist in the Office of the CTO at BEA, where he helps set BEA's technical vision and guide its execution. He came to BEA via the acquisition of WebLogic, where he was a pioneer in Java and Web application server technology. Prior to joining WebLogic, Benjamin worked on the original Java team for Sun Microsystems, where he helped create Java 1.0, 1.1, and 1.2.

CONTACT...

br@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

Remember Java applets? From a business perspective, the most interesting part of the show was an announcement that Netscape, then the hottest technology company in the world, was going to put support for Java in Navigator, then the world's dominant Web browser. Microsoft, many speculated, was dead. It had missed the boat on the Internet by trying to emulate AOL with MSN. Netscape, Sun, and the combined forces of the open standards community were going to bring the Windows "hairball," as Sun's CEO called it, down.

It has been not quite 10 years, but two things are clear: Microsoft and Windows are still here, and Java has become the most important thing to happen to computing in 25 years. The real revolution happened, and it happened on the server, not on the client. The advent of servlets, JDBC, and the worldwide adoption of J2EE as the standard for enterprise computing has changed the way the world does business. Clients have also changed a lot: whereas 10 years ago a typical software vendor would have written its front ends as a heavy-client application using VB or PowerBuilder, now more often than not they will write a Web front end. But make no mistake: the real revolution happened on the server side.

Java in the Enterprise

While the focus on Java enterprise technologies is often on technologies like Web services, messaging, or EAI, here we'll focus on a part of

the enterprise stack that is often overlooked: the Java Virtual Machine. At the heart of every Java application is a Java Virtual Machine, or JVM. A JVM is like a microprocessor, but implemented in software. A microprocessor is capable of executing a program written in machine code. That machine code is made up of machine instructions that are implemented by the microprocessor, in silicon. Likewise, a JVM is capable of executing Java classes that are made up of Java bytecodes, the equivalent of machine instructions. Unlike a microprocessor however, Java implements these bytecodes in software in a program called the Java Virtual Machine. This is what makes Java portability possible: you need to write a Java Virtual Machine for a specific architecture such as Macintosh OS X, or Microsoft Windows, or IBM's AS/400 (and associated chipsets) only once and then you can run any Java program, usually with no changes whatsoever. Java is now such a widespread and successful platform that most modern computing platforms, from handhelds to big iron, have at least one Java Virtual Machine available to them, usually several.

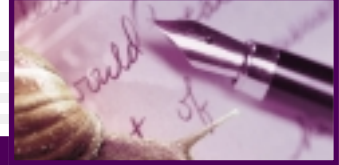
Because of an early focus on the client side, the first JVMs were focused on issues paramount to client-side performance, such as graphics rendering and event handling. Enterprise applications have different requirements. BEA Systems conducted a survey of the most important requirements in a Java Virtual Machine.

Reliability

Any time a server application goes down, possibly thousands of clients and often many other applications are adversely affected. This can create angry customers, lost time and revenue, and much anxiety for IT personnel. By contrast if a client application fails, the damage is almost always easily contained. Indeed, the most successful client platform of all times, Microsoft Windows, has never been particularly reliable. Not so on the server side.

Scalability

Client software typically has one user. Server software may have thousands of clients, both users and applications. It must be able to handle large loads, and more important, it needs to be able to grow and accommodate added load with additional software. For example, if you know that a server can handle 1000 users, you'll be able to handle 3000 users with 3 servers. This is called *linear scaling*.



Nondisruptiveness

Anyone who's been using Java for a long time knows about garbage collection pauses. The Java program simply stops, the disk grinds as the JVM reclaims unused memory, and the program then restarts. A nondisruptiveness requirement means that the infrastructure must never get in the way of the application. In the case of a JVM it means no long pauses. In a server environment, with potentially gigabytes of memory, this is a formidable technical challenge.

High Performance

High performance is like motherhood and apple pie: no one can argue with it and everyone wants it. But in a server environment this is especially important because it makes for a lower Total Cost of Ownership (TCO). TCO is a key concern of IT today, and when possible it is always better to use one machine rather than two. In addition, high performance is often a prerequisite to good scaling.

Client vs Server

Since enterprises require reliability, scalability, and performance from their server infrastructure, to provide these things we need to look at the specific needs of the applications that run on these servers. Server applications and client applications differ in fundamental ways. Figure 1 shows these differences.

Several things jump out right away. First, server applications make vastly greater use of threads and I/O, while client applica-

tions make greatest use of code compilation (just-in-time compilation, or JIT.) This makes sense because the role of servers is usually to perform many different tasks for many different clients (many threads), and to send back the result of these tasks (lots of I/O) while clients tend to do a lot of graphics-intensive work (lots of drawing and computation.)

Java Virtual Machines designed in the early days of Java, including the Sun Classic JVM, HotSpot, and HotSpot derivatives like the IBM JVM, were designed with a client focus. HotSpot, as its name implied, was the first JVM to focus on code compilation by detecting "hot spots," or sections of code that run often, and compile these sections on the fly - just-in-time compilation.

In 1999, students from the prestigious Royal Institute of Technology in Sweden attended a session on JVMs at JavaOne in San Francisco. Joakim Dalhstedt remembers his "Eureka" moment: "As we were listening to the talk, I thought to myself: this is all for the client - what if we were to design a JVM really optimized for the server?"

Appeal Virtual Machines AB was born shortly thereafter and in 1999, JRockit 1.0 came out. In 2001, BEA Systems hired the entire Swedish team and acquired JRockit. In a constant effort to ensure the broadest possible customer choice of platforms, BEA teamed up with Intel to make certain that a truly world-class JVM would be available for customers to use with Intel architecture servers.

From the very beginning BEA WebLogic

JRockit incorporated unique technology features for optimized server performance, such as multiple garbage collection mechanisms.

Different applications use memory in different ways. Because Java shifts the burden of memory management from the developer to the JVM, it is important that the JVM be as intelligent as possible about how to manage memory. One application for example may create lots of short-lived objects, while another may create mostly large, long-lived structures. One application may require 100 MB of heap while another requires 2 GB or even more. For JVM buffs, BEA WebLogic JRockit supports:

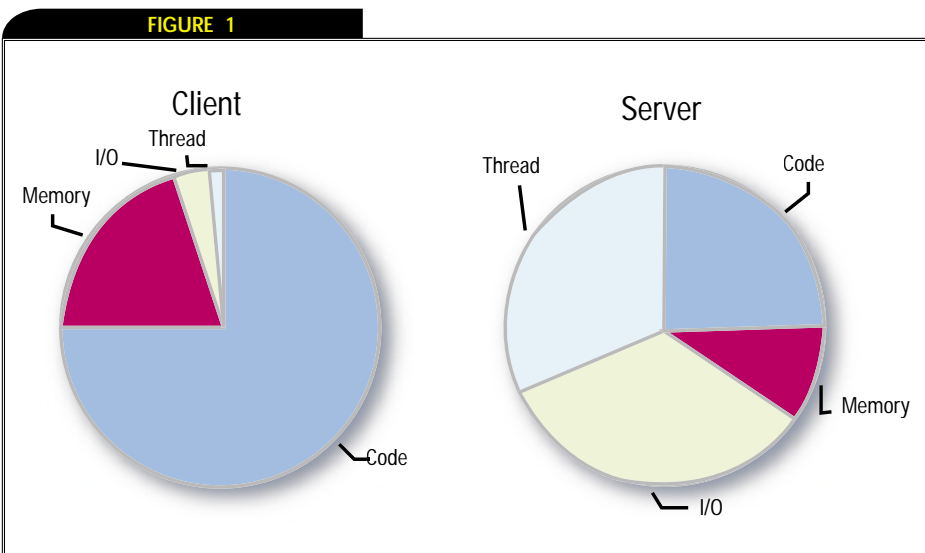
- Generational garbage collection
- Concurrent garbage collection
- Parallel garbage collection

Concurrent garbage collection is particularly exciting because it allows Java applications that can run without any pauses whatsoever.

JRockit also takes adaptivity further than most other JVMs. For example, based on how often a particular section of code is used it will spend more or less time optimizing it, thereby spending compiler cycles more effectively. Another area where the JRockit team is hard at work is GC adaptivity. Today the type of garbage collection must be determined manually, through the usual process of application performance testing. The JRockit engineers are currently hard at work on a mechanism that will automatically determine which garbage collection scheme is optimal and set it without user intervention.

Management, Management, Management

One of the ongoing frustrations of developers and IT operations personnel alike has been the black-box nature of JVMs up to now. WebLogic JRockit goes a long way toward addressing some of these grievances. The BEA WebLogic JRockit Management Console lets users find out a wealth of information about what goes on in the JVM, such as detailed memory and thread usage information. In addition, the user can configure the JVM to send warnings when suspicious or critical conditions are reached, all entirely configurable. Most important, this monitoring capability comes basically free of charge.



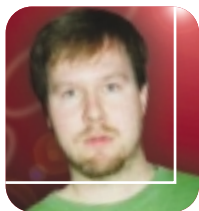
Differences between server and client applications

- continued on page 47

BEA WebLogic Integration 8.1 from BEA Systems

MOVING AHEAD OF ITS COMPETITORS

Reviewed by Jason Snyder



Complex Business Process Management (BPM) solutions involving workflow creation, enterprise resource access, and real business tasks can quickly become unmanageable. The workflows can grow into giant, hard-to-follow decision trees, the developers have difficulty transforming the requirements into actual code, and sub-processes intended to be simplistic become overly complex applications. BEA has greatly simplified this effort with the new release of WebLogic Integration 8.1. I recently created several sample processes with the beta release of this product and was very impressed with the results.

WebLogic Integration is incorporated into the WebLogic Workshop framework. This allows a consistent means of development and allows its users to become familiar with only one tool when writing J2EE applications, portals, and/or BPM solutions. The benefits of this “unified platform” approach cannot be overstated. The intuitive user interface provides a one-stop shop for modeling business solutions, analyzing them, and then creating BPM results that access enterprise applications

and allow true user collaboration. Using the tool and some examples provided by BEA, in less than one day I created several business processes that performed complex conditional logic, data transformation from XML to Java, e-mail notification based on user role, and asynchronous message communication. I barely scratched the surface of what this tool can do.

Creating a Workflow

My first goal was to create a simple workflow that would provide two separate paths for processing information. The Workshop editor allows graphical creation of workflows (see Figure 1). While the user is adding various components, the source code is written and accessible on a separate tab. Workshop generates source code and annotations to reflect the process logic, using the Process Definition for Java (PD4J) specification that is in the standards process (JSR 207). The Workshop editor keeps the code and graphical representation in synch.

Creating the workflow is easy. The palette on the bottom left-hand side of WebLogic Workshop has a number of clear alternative nodes that can be added to the workflow sim-

ply by dragging and dropping. The workflow nodes are plentiful enough that any workflow action I could identify could be performed, and yet are clear enough that it's very easy to identify the necessary node.

The workflow representation is very clean looking, and is presented in a manner that can be followed easily. It's not beyond the realm of software development to envision a situation where the business user could create the initial flows for developer augmentation. Adding conditional operations was very straightforward. I would recommend though that you identify the basis for each conditional statement in the condition name in order to provide the easiest means of workflow readability.

For passing data into my workflow, BEA has focused on providing a real means of easily including XML in application development with their XMLBeans. The concept behind XMLBeans is to separate the complexity of XML creation from its usage. XML objects can be accessed and manipulated easily through the use of a GUI. WebLogic Workshop handles the parsing by having each XMLBean act as a Java interface to the XML Schema. Developers can focus on the business logic and not focus on the drudgery of XML creation and access.

Another key aspect regarding XML and WebLogic Integration is the ability to transform this data. XML Schemas can be converted into other XML Schemas or non-XML Schemas, or XML Schemas can be created from other formats. Complex XML mappings can be constructed without any knowledge of XSL, XQuery, or any XML mapping APIs.

Mapping the data across schemas involves linking the two values together, and necessary mathematical or procedural operations on the data can easily be added. WebLogic Integration 8.1

BEA Systems, Inc.
2315 North First Street
San Jose, CA 95131
800 817-4BEA (U.S. toll free)
408 570-8000

Web:
www.bea.com

E-mail:
sales@bea.com

Download information:
<http://dev2dev.bea.com/subscriptions/index.jsp>

Requirements
BEA WebLogic Integration 8.1 comes with a clustered instance of BEA WebLogic Server 8.1 and development licenses of BEA WebLogic Workshop. BEA WebLogic Integration 8.1 requires BEA WebLogic Server 8.1.

also allows users to design complex mappings such as many-to-one (e.g., aggregation), one-to-many, or many-to-many. BEA calls this an XQuery Transformation Mapper (see Figure 2).

The data transformation mapping could be improved by allowing it to guess some mappings, particularly when they share the same name. It is important to test the mapping and compare the result with what you expected. A successful XQuery Statement execution does not guarantee that everything was accurately mapped correctly.

For processing a variety of operations, WebLogic Workshop provides “Java Controls.” Think of them as a series of keys into most of the technical doors that must be accessed. Existing data sources, systems, applications, and other workflows can all be accessed. A control for accessing a database can be created. If JMS access is required, a Java Control could be created. I created one to process e-mail.

As with any tool, some Java Control functionality was not immediately identifiable, and many of the controls should be played with to identify their potential usage. Custom controls can be created as well, so almost all operations could be accessed from WebLogic Workshop through a Java Control.

The fact that workflows themselves could be used as controls also provides flexibility. Other workflows, Web services, or portals could directly access these workflows. Integration with existing functionality becomes an almost no-thought task.

Integration with the application server is seamless. Testing can be performed within the Workshop environment.

My next effort involved creating an asynchronous JMS message. This is a realistic situation, as many workflows do not involve the synchronous communication found in many applications.

When your bosses assign you a task, they don't stand over your shoulder while you complete it. Well, at least not all of the time.

This process involved the creation of two workflows: one for sending and one for receiving. A JMS Java Control was created in minutes, and the control was added to the workflow node by dragging and dropping. The mechanism for involving JMS could not be any easier.

My next effort involved more real-world simulation by adding human users to the workflow effort. WebLogic Integration provides a Worklist subsystem that allows tasks to be assigned to users, and to send and receive these tasks via a simple user interface. This involved a Java Control and was intuitive to operate. Users can be specific or part of a group. Role assignment was a little complex, and I think that any developer using that functionality should allow for some additional time to come up to speed in that area.

Combining all of the functionality to add a simple application front end to the workflow and produce an e-mail output was not as complex as one you fear. It is worth noting again that because the WebLogic Workshop interface is used consistently for WebLogic Integration, it is easy to identify how new untried functionality is implemented. Since WebLogic Portal uses the same interface, a developer's learning curve is limited, and most functionality involves a learn-once method of usage. I was surprised at how often my “guesses” at how a feature is implemented were correct.

The administration console was browser-based and seemed to provide all necessary functionality. A repository stores runtime information, which allows for later detailed analysis.

Exception handling was easy to include, and more applica-

tion-specific means of exception handling appeared to be easy to add.

Conclusion

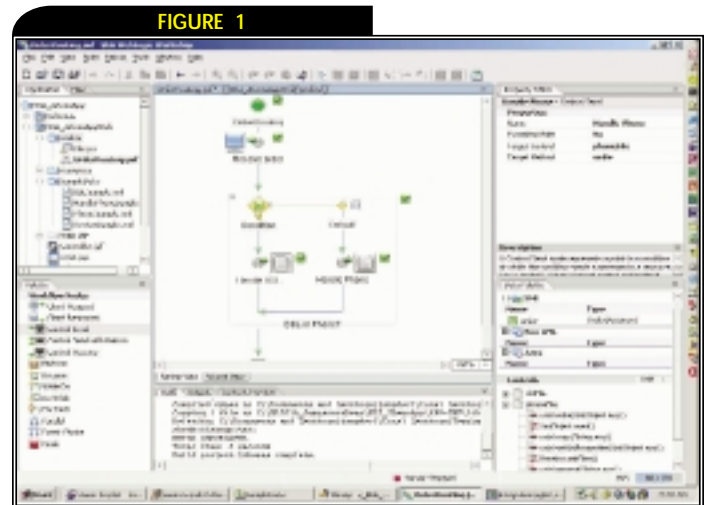
BEA WebLogic Integration 8.1 provides an intuitive and powerful means of developing integrated business processes. Combining these elements is always difficult, and BEA deserves high praise for the consistency with which they have succeeded across WebLogic Platform 8.1. Each component of this platform is remarkable, and the integration across the platform provides functionality that is years ahead of the alternatives. 🍷

AUTHOR BIO

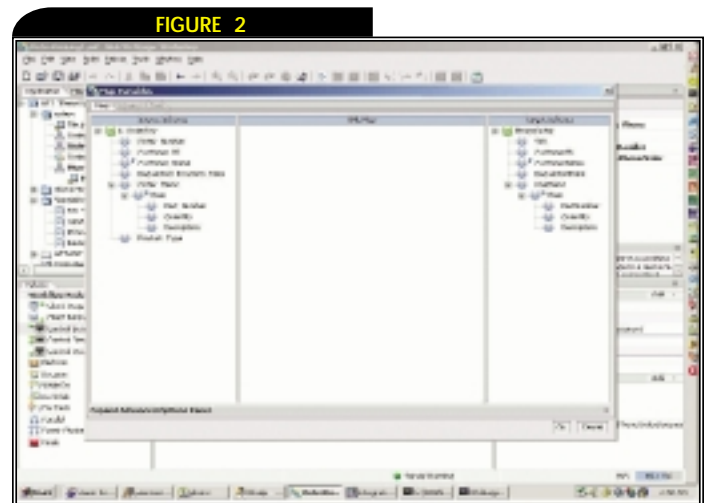
Jason Snyder is an architectural expert for CSC Consulting in Boston, and has served as the lead architect for several J2EE development projects. He has over 10 years of experience in software development, OO design, and application architecture.

CONTACT...

jasonsnyder@townisp.com



Visual Representation



XQuery Transformation Mapper



TRANSACTION MANAGEMENT

In most large-scale “mission critical” systems, high on the list of requirements is resistance to failure. With the world living in fear of violent destruction post 9/11, it is more common for the definition of “failure” in this context to be the loss of a whole data-processing facility.

Freedom, Disasters, and Getting Something for Nothing

HOLDING THINGS TOGETHER

BY PETER HOLDITCH



AUTHOR BIO

Peter Holditch joined BEA as a consultant in the Northern European Professional Services organization in September 1996. He now works as a presales architect in the UK. Peter has a degree in electronic and computer engineering from the University of Birmingham.

CONTACT...

peter.holditch@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

Taking the provision of a full failover site into consideration provides some good food for thought in terms of the best way to architect a solution.

Having decided to split data processing over two sites, it then needs to be determined whether the sites operate in a dual-running mode, whereby both share the transaction load during normal operation or in a master/backup configuration where the processing all occurs on the “master” site, and entirely fails over to a backup site should disaster strike. In these parsimonious times, it is usual to want dual operation to cut down on the solution cost, so lets think about what parts of J2EE we need to employ to get dual sites running in parallel, with each having the capability to fail over to the other in disaster situations.

Before starting down the road of technology solutions, there is an architectural choice to be made, a choice which in its turn may be driven by further business requirements: namely, how “synchronized” does the data on the two sites have to be during normal operation? Is it acceptable for the mirrored data on the two sites to be inconsistent, and if so for what duration?

If the two sites must be completely synchronized, then the good old transaction manager comes into play – you make all your database updates twice in the context of a JTA transaction,

and the two databases are guaranteed to be synchronized to within a whisker of time. Of course, when disaster strikes and one of the databases becomes unavailable, one of the dual updates must be dropped – otherwise the whole system won’t be able to function – how can a transaction commit if only one of the two resources it touched is available? So, you code some logic such that the application ceases to send the remote updates when the remote site is unavailable. Of course, someone digging a trench through your fiber-optic cable, rather than a real disaster, may cause the apparent disappearance of the remote site. In that case the systems are said to be partitioned, and after some interval the two (both of which have been gaily processing on the assumption that the other has died) will be reunited. At this point, you have a bit of a headache since the databases are now inconsistent in ways that need detecting and reconciling.

Clearly, what I have described is a lot of design – and consequent lines of code – of an infrastructure nature. I guess that’s okay, since the business requirements are also of an infrastructure nature. It’s just as well that you had the transaction manager at hand, though – just think how much more you’d have had to worry about without it!

Usually, because of the onerous nature of maintaining such tight levels of synchronization between sites, the requirements relax somewhat and some window of inconsistency is tolerated between replica databases on different sites, with updates on one propagated to the other asynchronously. In this scenario, the JTA transaction manager provides transactional consistency not between two databases, but between the local database and a JMS queue, which will be used to propagate updates to the remote site as and when it can. In this case temporary connectivity losses between the two sites are less of a problem because of the tolerance built into the requirements of data being out of synch. When the connection is lost, messages queue locally and when the network comes back transmission resumes – putting much less pressure on the management of inconsistency between the databases. (Of course, in practice the lower pressure is a direct result of the looser consistency requirement.) The good news is that the transaction manager is still there to ensure that updates are only published via the queue once they are committed to the local database.

In noncatastrophic failure cases, it is likely that the database that was running on the failed system will need to be failed over to the system that is still running (maybe we’re talking about a CPU



failure now, not total destruction of a facility). This will clearly require that the physical disks that the database is stored on are available on the primary and the backup machine – in the first instance, the database engine will need to be restarted on the secondary node so that it can pick up managing the data again after recovering its state. Likewise, the storage backing up the JMS queues will need to be failed over too. It is at this point that we must count the cost of using JTA – it isn't magically giving you something for nothing – to make sure updates between databases, or between queues and databases, are consistent. The transaction manager itself needs to keep a persistent record of what is going on; this record is held in the transaction log. Therefore, to do a failover for the queued case, not only do the persistent stores behind the queues and databases need to be moved across and recovered, but the transaction log needs to be moved too. Moving it is achieved the same way you moved the database files – dual ported disks, storage area networks, whatever you have at hand. Once this is done, the databases and queues need to be recovered and the migrated transaction log needs to be restarted to allow in-flight decided transactions to complete, while aborting and cleaning up after those that did not get to the point of being decided. WebLogic's console provides for the administrator to migrate the transaction recovery service from a failed server to a healthy one; doing this is what completes the in-flight transactions according to the content of the migrated log file.

“What is certain is that somewhere, at some level in your solution, whatever shape it takes, is a transaction manager (or something that smells like one) holding things together”

Conclusion

In conclusion, there are two observations to be made here. The first is that a complete solution to this type of requirement requires design and implementation at multiple levels – the storage hardware, DBMS, application server, and application logic all need to work together to provide support for graceful failover. The second point, which to

some extent follows from the first, is that (as with all architectural design) you have a choice as to what level to implement things at. A hardware clustering solution could fail over a disk and restart all the application-level facilities, but these tend to be regarded as expensive and operate most easily in the master/hot standby mode. Databases offer replication techniques too; however, they suffer from the same laws of physics that I discussed above. Changes are replicated asynchronously, allowing for windows of data inconsistency, with the added problem that the code you need to implement to resolve conflicting updates will be tied to the particular database engine you have – this kind of thing is way beyond what standards in the database arena specify – and also, since the replication happens behind the scenes, diagnosing problems and determining the application's desired state from a set of inconsistent logs, etc., requires extremely deep knowledge of database internals, if it's possible at all.

Finally, you can go with the application-server approach, which has the benefit of being portable across application servers, since it's based on standard programming interfaces, and also provides transparency as to what is happening with the replication since it is all “above the covers.” The downside of the application-server approach, arguably, is that the transaction log is yet one more thing to manage and fail over. To that, I'd say that you get what you pay for in terms of application manageability and transparency.

In reality, any failover solution of this nature will be designed to use different capa-

bilities provided at different levels in the architecture. The choice of what to do at what level cannot be made in a generic way since all the detailed requirements of any given application will vary so widely. What is certain is that somewhere, at some level in your solution, whatever shape it takes, is a transaction manager (or something that smells like one) holding things together. ●

ATTN: Developers

STEP UP
to the mike
and be...

Go to
<http://developer.sys-con.com>

HEARD!

**Calling Sleek
Geeks Everywhere!**

Make sure you have your
finger on the pulse of
i-Technology...bookmark
<http://developer.sys-con.com>
today.

i-Technology

News

i-Technology

Views

i-Technology

Comment

i-Technology

Debate



© COPYRIGHT 2003, SYS-CON MEDIA
WWW.SYS-CON.COM

**SYS-CON
MEDIA**

WebLogic Support

REACHING OUT TO YOU
IN YOUR HOUR OF NEED!

BY APURB KUMAR



AUTHOR BIO

Apurb Kumar is a developer relations engineer in Backline WebLogic Support at BEA Systems. He has more than 10 years of experience, starting with real-time programming, moving on to databases, and finally Java development. Before moving to BEA Systems, Apurb consulted for companies such as Charles Schwab, AllAdvantage.com, and Holland Systems.

CONTACT...

apurbk@bea.com

REPRODUCED WITH PERMISSION FROM BEA SYSTEMS.

When I first transitioned to BEA WebLogic support, I had some premonition of what lay ahead, but little did I realize the daunting task of working on third-party code under high-pressure conditions.

Unlike development, where most of the time you wade through your own piece of code, software support becomes more complex as you work on system-down and critical situations, day in and day out.

This article will share some of the experiences I've had with customer cases. It includes some very basic but useful tips that WebLogic developers can utilize to help themselves, before opening a case with support.

- WebLogic Knowledge Base
- Proper environment
- Confirm J2EE specs
- Beware of third-party software limitations
- Confirm platform configuration
- Create a test case

WebLogic Knowledge Base

BEA wants to provide you with the maximum resources and assistance, in the least amount of time. It is for this purpose that we have the AskBEA Web site, where you can post questions and search the database for the nearest news-group discussion/solution/documentation results surrounding your search criteria. Using this tool can be very beneficial in the long run, and can save you tremendous time and energy, as you can find the solution to your posting on our Web site rather than going to the trouble of opening cases and later finding that the answer was readily available. Why reinvent the wheel?

Proper Environment

One of the most common scenarios I've come across is, "Our application was working fine in QA but when moved to production, it boomeranged!" Oops! "There's many a slip between the cup and the lip!" The first suspects to look for in these situations are the environment settings, especially the classpath. Check to confirm that you are on the same patch levels in production as in QA. The order of the patches in the classpath does matter, as the system class-loader will load the classes in that sequence. If you are running with multiple patches, the contents of the JAR files should not have the same classfile names. If they do, that means the patch that comes later in the classpath would have no effect, as this class won't be loaded. This is one of the important reasons why you should provide the current patch levels to support should the need arise.

Another quick check would be the configuration settings between the config.xml for the QA version and for the production version. For checking performance metrics, make sure that you are talking about the same hardware and approximately the same kind of load as when your application was tested in QA.

Confirm J2EE Specs

Getting Assertion Errors? NullPointerExceptions? While these errors are rare and might mean that the exception handling was improper at WebLogic's end, it is also very likely that the exception was thrown as something happened that was not expected by the WebLogic server. But was that "happening" allowed by the J2EE specification? Before you grab the phone to call support, if possible check out the specification of the J2EE sub-component that you're working on. I remember a case in which the client was doing extensive thread manipulation inside the EJB class, and he was running into all kinds of issues. He was pretty frustrated with the way 'WEBLOGIC' would throw exceptions all over the place, and had spent considerable time making sure that he had not done anything wrong syntactically inside his application code. It turns out that he didn't have a bad line of code, he had a flawed design. The EJB specification prohibits spawning user-defined threads inside the bean class. Hence, it saves a lot of joint effort to go through the specification before opening a case.

Third-Party Software Limitations

This is more of a “biting off more than you can chew” situation. Or more practically, placing a 1000W bulb in a 100W socket! I have this example. A customer wanted to create Multipools on top of two connection pool. Each connection pool was created on top of two parallel database servers. The customer wanted to have a Transparent Application Failover (TAF) so if there was an in-flight transaction in progress when a database hardware failure occurred (tested by powering off the database), the transaction would be completed by the other server. Well, after plenty of research, and after opening a case with the database company, we found that TAF wasn't being supported by the drivers used by the parallel servers. Hence, multipools/connection pools, which are wrappers around the actual connection, can't help much in this regard.

As far as third-party utilities are concerned, it's quite possible that you may know more about it than we do. Hence, if you're using third-party tools in conjunction with WebLogic it makes sense to double check that the third-party utility doesn't have a limitation. Furthermore, in the same spirit, it would probably be easier to isolate whether the problem is on WebLogic's end by removing the third-party part and retesting your application.

Confirm Platform Configuration

A lot of problems and issues arise if development is done without heeding the recommendations of supported platforms and configurations. I have often found that customers are on unsupported configurations. If the customer is in development it's easy to remedy the problem. Enter production, and it becomes an uphill task! I remember an instance in which the customer was having tremendous problems in their production environment as managed servers were dropping out of the cluster. We narrowed it down to the fact that garbage collection was running for too long (more than 230 seconds), which resulted in missing the server heartbeats. Guess what? The customer was on an uncertified JDK version. They could not move their JDK version to a supported configuration, as it would result in hundreds of compile-time errors in their application (which was heavily dependent on the new JDK API).

In the same light it would be prudent to add that it pays to check not only the JDK but also the supported OS version, and whether OS-level patches are required on top of everything. You might also want to check the TCP-level settings mentioned for the particular OS.

Create a Test Case

While developing your application it is common to start getting stack traces in the weblogic.log files. Once you are certain that you are doing everything per the J2EE specification and that the code is correct, the next step is to get a reproducible test case. Remove the accessories (we certainly don't want any proprietary stuff passed around), and narrow it down to the problem at hand. This is a vital and key point. You know your application better than anybody else! A narrowed-down version of the problem at hand helps us to focus and debug the problem more easily. In all my experience, I have found test cases to play a major role in the turnaround for a resolution.

Baby Steps

You've encountered a problem and opened up a case with WebLogic support. Now what? The DRE (developer relations engineer) is working with you. Meanwhile, since you are still in development, you encounter another issue. Well, instead of putting this problem in the same case, open another one. This helps the DRE focus on one problem per case with you. The next thing to watch for is the communication. This holds another vital key to the entire process. Watch out for situations in which the DRE thinks he/she is expecting something from you, and you think that he or she should already be working on getting a workaround/fix.

Conclusion

All's well that ends well! In our case, we not only want the end to be better, but also the entire process to be a pleasantly memorable one. The bottom line is that we want to create solid, healthy relationships with our customers – *you*. At the end of the day what counts is to be a part of the happiness that you gain when your case is resolved. 🍌

FROM THE OFFICE OF THE CTO

– continued from page 41

When the user turns off metrics collection, there is virtually no overhead. And to make it all even easier, the console is able to manage multiple JVMs at the same time.

One of the shortcomings – overcome in the most recent version of BEA WebLogic JRockit, 8.1 – is the lack of support for JVMPI, the standard profiling interface for JVMs.

Fastest JVM Anywhere

JRockit is not only exceptionally well architected, robust, and manageable, it is also the fastest JVM available in the lab as well as in the field. BEA WebLogic JRockit consistently received the highest marks in industry-standard benchmarks such as SpecJBB and SpecJVM, as well as more specialized benchmarks such as SPECjappserver. That success, says Arvind Jain, product manager for WebLogic JRockit, translates into real TCO gains: “In today's budget-constrained IT environments, customers are not only placing a high value on raw performance but also price-performance.”

Customers seem to agree. At BEA's eWorld user conference in February 2003, Bank of New York presented their most recent efforts to test their existing infrastructure on JRockit and Intel-based servers. They found dramatic cost savings, on the order of 70% in some cases.

The moral of the story is that while some of the most exciting developments, like XML, EJBs, and Web services happen at the top of the stack, some of the most important ones happen deep down, in the JVM.

JRockit is available at www.jrockit.com. 🍌

SAVE 17% OFF

DEVELOPER'S PowerBuilder Journal

12 Issues for \$149⁹⁹

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

- New PowerBuilder features
- Tips, tricks, and techniques in server-side programming
- DB programming techniques
- Tips on creating live PowerBuilder sites
- Product reviews
- Display ads for the best add-on products

That's a savings of \$31 off the annual newsstand rate. Visit our site at www.sys-con.com/pbdj/ or call 1-800-303-5282 and subscribe today!



Improving Performance, Avoiding Hung Threads, and Keeping It Simple

KNOW THE RULES BEFORE YOU BREAK THEM

BY LEWIS CIRNE



AUTHOR BIO

Lewis Cirne, founder and CTO of Wily Technology, invented the company's core patented Java Agent technology. As Wily's CTO, Lewis takes a leading role in the future technological development of the company and its products, with the goal of extending the services offered by Wily to customers deploying high-performance e-business solutions.

CONTACT...

asklew@sys-con.com

As always, I invite you to send an e-mail to asklew@sys-con.com if you have any performance-related questions about JVMs, Java applications, WebLogic Server, or connections to back-end systems.

Q. What are some quick steps I can take to improve CPU utilization?

A. If your application is experiencing serious performance problems, I recommend that you perform a thorough analysis to determine the root cause. But if you're looking for some quick ways to optimize application performance, here are a few tips that may help.

- **Distribute your JSPs.** Much of the CPU burn for many applications is consumed by parsing back-end data up into data structures for hand-off to the application's view (JSPs typically). If the CPU is burned in that part of the application, then chances are very good that it also gets burned while pulling apart these data structures when building the JSP itself. Try distributing your JSPs into a container separate from the business logic. As a result, more CPU will be available to help out with back-end data requests. Better yet, you'll understand whether the "front" or the "back" of your architecture burns more resources, so you can tune and/or add resources accordingly.

- **Review the semantics of your application's back-end connection usage (JDBC, JMS, etc.).** Make sure all calls to distributed remote systems are wrapped with sensible timeouts (i.e., if a call hasn't come back in 15 or 30 seconds when it normally takes less than 5, then send a sensible error message back to the user). This helps you avoid the risk of hanging the entire application server due to a single failed resource.

- **Reduce the amount of data offered in search results.** If your application returns search results, make sure it hands back less data at a time – not just to the user but to the WebLogic Server instance itself. If your application is fast, they won't mind clicking again. Large amounts of data can create a "pig in the python" problem throughout your entire application architecture.

- **Reduce unnecessary logging.** As I mentioned in an earlier column (*WLDJ*, Vol. 2, issue 7), while logging is a great analytical tool, too much logging can cause excess CPU utilization. Consider putting a dynamic switch into your application that will allow you to redirect the logging to /dev/null whenever it is unnecessary.

Additionally, when logging, you can use the features available in JDK 1.4 to select the appropriate level of granularity.

- **Measure, measure, measure.** This final point is very important. There is no substitute for gaining a complete understanding of where your application is using resources and why. If you understand the answer to those questions, you'll know what it takes to improve your performance.

Q: What are some good defensive programming techniques I can mandate for my developers to avoid a "hung thread" or "hung server" in production?

A: In another column (*WLDJ*, Vol. 2, issue 6), I discussed finding the causes of stalled threads. Here are some techniques that can help you avoid them altogether. In most cases, a "hung thread" is one that has blocked on contacting a back-end system for data needed to dispatch its request. The most common of these is a database, in which case it is probably being contacted via JDBC. If your application is passing messages, it may be a JMS implementation. If it is a proprietary or legacy system, it might talk only a specific protocol over TCP/IP. The list goes on.

The key is to make sure that the application knows what to do if the resource it needs is just not available. Most of the time, this isn't a problem because the code for timing out sockets to the remote DB server is all in the JDBC driver.

When you have to handle timing out a connection yourself, the situation is significantly more complicated. David Reilly wrote a very informative article on handling network timeouts entitled "Simple Handling of Network Timeouts," which can still be found on the Web.


There is no replacement for understanding how your transactions will behave at the code level when remote resources become unavailable. There are, however, a number of parameters in the various layers of software supporting your application that may be tuned to obtain better failure patterns right away. WebLogic Server itself includes a number of parameters for tuning how the EJB container transactions are

timed out (<http://edocs.bea.com/wls/docs70/ejb/reference.html#1072610>).

The moral of the story is: if you're struggling with hung threads and suspends in your application, or you want to avoid struggling with them, spend time thinking about how each and every one of the connector components in your application will behave when the back-end resource on which they rely disappears.

Q. Of the application architectures you've seen, which have been the most successful?


A: When it comes to matters like this, my philosophy has always been to keep it simple. In my experience, the projects that usually end up on time and under budget are those that have been completed in the most straightforward way possible.

Does this mean that you can ignore the principles of object-oriented design? No, but there may be times when you have too much of a good thing. Most architects of object-oriented applications will rightly object to two different teams creating their own versions of a thread pool. Doing this will result in a larger, less maintainable code base; bugs in one version will not necessarily get fixed in the other. But what about the logging strategy adopted for an application? One application may need logging for debugging and performance analysis. Another may need it to audit every transaction, perhaps for security purposes. Insisting on a unified logging framework for an entire enterprise might result in a logging approach that doesn't necessarily make sense for each application. For many applications, having a separate servlet for every important application use case, a small library of business logic that is not necessarily EJBs, and a JSP for each customer view of the Web application just might be exactly what you need. The bottom line is, keep it simple but be sensible; don't break the rules unless you know them first. 


WLDJ ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
BEA Systems	http://dev2dev.bea.com/useworkshop	800-817-4BEA	11
ColdFusion Developer's Journal	www.sys-con.com/coldfusion	888-303-5282	24
Confluent Software	http://www.confluentsoftware.com	408-523-9000	3
Hewlett-Packard	http://devresource.hp.com/wldj.html	800-752-0900	2
Informatica	www.informatica.com	800-653-3871	15
Intel	www.intel.com/ad/bea	800-628-8686	21
Isavix Corporation	www.isavix.net	866-472-8849	29
Java Developer's Journal	www.sys-con.com/java	201-802-3021	25
LinuxWorld Magazine	www.linuxworld.com	888-303-5282	39
Panacya	www.panacya.com	877-726-2292	51
PowerBuilder Developer's Journal	www.sys-con.com/pbdj	888-303-5282	47
Quest Software	http://java.quest.com/jcsv/wldj	800-663-4723	9
Quest Software	http://java.quest.com/jprobe/wldj	800-663-4723	17
Web Services Edge West	www.sys-con.com	201-802-3069	33-38
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	28
Wily Technology	www.wilytech.com	888-GET-WILY	5, 52

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser.


LOOK WHAT'S COMING NEXT MONTH

- 


Enterprise Portals Integration and the Enterprise Service Bus

While BEA WebLogic Platform 8.1 is a robust platform capable of handling many application scenarios (including portals), certain types of portal integration scenarios require a new form of infrastructure, known as the Enterprise Service Bus.
- 


SPECjAppServer2002 Performance Tuning

In a top-down, data-driven, and closed-loop approach to performance tuning, we touch on key advantages of BEA WebLogic that improve the performance of J2EE workloads.
- 

Diagnosing Application Failures in BEA WebLogic


Having logging filters installed and running on your production servers and your centralized logging system, combined with custom logging instrumentation, will significantly reduce the amount of time required to diagnose serious application problems.
- 

Transactional Web Services

Transactions between enterprises, and support for complex business process execution, have driven consideration of technologies that relax some of the ACID transaction guarantees that programmers have used for many years.
- 

Be Paranoid, Be Very Afraid!

Don't use NotSupported, Never or Supports for your container-managed transactions; otherwise, you'll store up a whole load of grief for yourself.



News & Developments

Anexinet and BEA Systems Announce Partnership

(Philadelphia) – Anexinet Corporation, a regional solutions provider, has entered into a partnership agreement with BEA Systems. By partnering with Anexinet, BEA will now be able to extend its WebLogic Platform to mid-market companies on the East Coast, specifically Pennsylvania, New Jersey, Delaware, and upstate New York. The high demand for the WebLogic Platform will now also be met by Anexinet's knowledge of the BEA product suite and their educational and professional services.



www.anexinet.com

SAP Users Choose BEA WebLogic Server

(San Jose, CA) – The BEA WebLogic Server has been honored with the 2003 SAP FAQ Members' Choice Top 25 Tools and Technologies award. Sponsored by the SAP FAQ, the leading peer-to-peer technical support network for SAP professionals, the award honors the top 25 products that have had the greatest positive impact on SAP R3/mySAP implementations.

BEA WebLogic Server is the core of the BEA WebLogic Enterprise Platform, and BEA's flagship application server, which earned the company's reputation for providing incomparable reliability, availability, scalability, and security. It can help customers shorten development cycles, increase operational efficiency, automate



business processes, and adapt to changing business conditions – all while reducing overall costs. www.sap.com, www.bea.com

IT Market Research Firm Reports BEA Tied for Number One Ranking

(San Jose, CA) – BEA Systems has been placed as the joint leader of the application server software platform market with a \$100,000 lead on its nearest competitor, as measured by worldwide license and license-related services revenue, according to analysis recently released by IDC. BEA holds a 16-percentage-point lead over its nearest competitor in the market for application server software on the Unix operating environment; and a 16-percentage-point lead over its nearest competitor on the Linux operating environment. BEA continued to grow its share year-to-year, demonstrating further consolidation around the leaders in this market.



www.bea.com

Siperian Expands Alliance with BEA

(San Mateo, CA) – Siperian Inc., an innovator in customer data management software solutions, has announced an alliance with BEA Systems, Inc., to improve the customer data management experience for enterprise users. BEA has identified Siperian as a key partner for its Customer Experience Management (CEM) Business Solutions. CEM solutions leverage and extend an organization's existing investments in customer relationship management (CRM) systems to achieve higher customer satisfaction.

Built on BEA WebLogic Integration, the Siperian



Activity Server can provide a complete customer view across

the enterprise by integrating, reconciling, and organizing customer data and content across multiple sources in real time. In addition, the joint solution offers customers the ability to rapidly deploy innovative customer management strategies configured specifically to their business needs and deliver relevant information directly to the business user.

www.siperian.com, www.bea.com

Quest Software's JClass ServerViews 4.0 Eases Development

(Irvine, CA) – Quest Software, Inc., a provider of application management solutions, has announced the release of JClass ServerViews 4.0, a set of server-side Java components that now features new design tools that simplify the development of professional-quality charts for J2EE and server-side Java applications. JClass ServerViews enables Web content creators to visually design high-quality charts without having to know XML or Java programming.



The new version includes JClass ServerChart and JClass ServerReport components, and integrates with BEA WebLogic. www.quest.com.

Wakesoft Launches New Architecture Platform

(San Francisco) – Wakesoft has released the Wakesoft Architecture Platform, Version 4, which allows project teams to mitigate the risk of delivering J2EE projects while ensuring greater services orientation, visibility, and control aligned with strategic architectural goals.

The Wakesoft Architecture Platform includes a new ver-



sion of the Wakesoft Architecture Server and adds three new components that in combination enable customers to mitigate risk of J2EE project delivery, while preparing for their larger SOA strategic initiatives. www.wakesoft.com

AdventNet Releases ManageEngine Applications Manager 5

(Pleasanton, CA) – AdventNet, Inc., a provider of standards-based management software for applications and networks, has released ManageEngine Applications Manager 5, which proactively monitors and manages the health of business-specific and mission-critical Java and J2EE applications using standards-based technologies such as JMX. The console monitors all components and an application deployed on the BEA WebLogic Server and presents them in intuitive dashboards and graphs.



The new release features the ability to view user-defined metrics by generating custom consoles and dashboards to display information on the user's preference.

www.adventnet.com



Panacya

www.panacya.com

Intel

www.intel.com/ad/bea