INTEGRATION • PERFORMANCE • BPM • PERSONALIZATION

www.SYS-CON.com/WebSphere

WebSphere® JOURNAL

The World's Leading Independent WebSphere Resource

NOVEMBER VOLUME 2 ISSUE 11

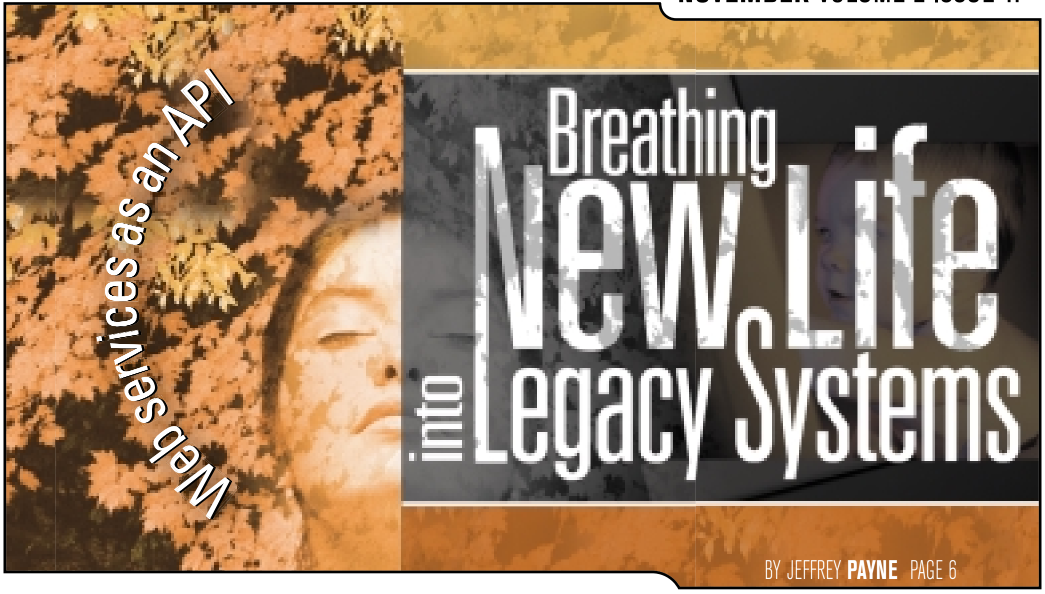
FROM THE EDITOR
Outsourcing: Magic Bullet or Dirty Word? It All Depends on Your Perspective
BY JACK MARTIN • PAGE 4

WEBSPHERE STUDIO
Teamwork in WebSphere Studio
BY ADAM TERRENZIO • PAGE 36

FINAL THOUGHTS
On Protecting U.S. IT Jobs
BY ADAM KOLOWA • PAGE 50

WEBSPHERE NEWS
PAGE 48

DISPLAY UNTIL JANUARY 31, 2004
$8.99US $9.99CAN



Breathing New Life into Legacy Systems
BY JEFFREY PAYNE PAGE 6

One Access Point to Rule Them All
The Front Controller pattern simplifies integration challenges
BY LLOYD HAGEMO AND RAVI KALIDINDI PAGE 14

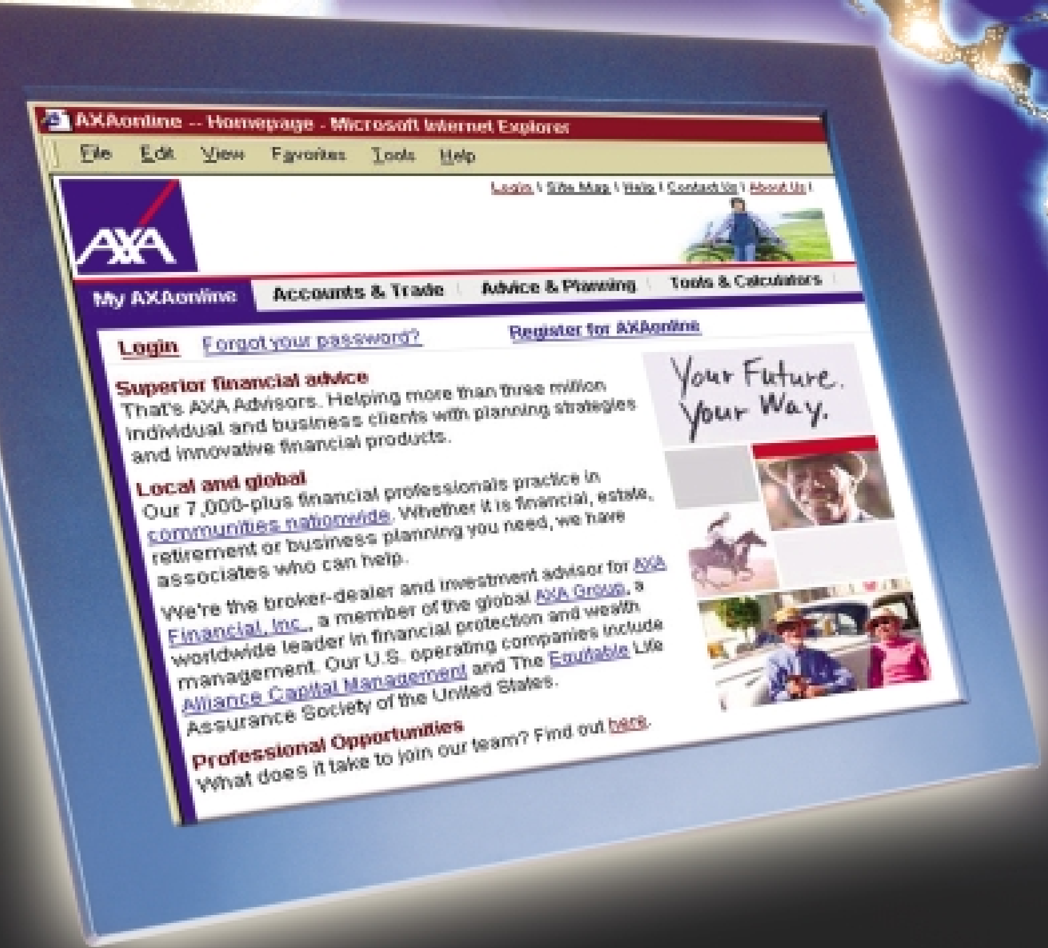
Managing the WebSphere Platform
Getting an end-to-end view of your applications
BY MARINA GIL-SANTAMARIA PAGE 18

Understanding Straight-Through Processing
A technical overview
BY JAMES LIDDLE PAGE 22

Implementing J2EE/.NET Interoperability Using WebSphere MQ
Part 2: Putting theory into practice
BY BORIS LUBLINSKY AND DIDIER LE TIEN PAGE 26

Best Practices for Using XSLT in WAS Applications
Use of XSL offers many strategic advantages
BY HARVEY GUNTHER PAGE 40

Getting a Handle on Your Customers
The better you know your customers, the more you can sell them
BY E. KENNETH NWABUEZE PAGE 46



Superior financial advice
That's AXA Advisors. Helping more than three million individual and business clients with planning strategies and innovative financial products.

Local and global
Our 7,000-plus financial professionals practice in communities nationwide. Whether it is financial, estate, retirement or business planning you need, we have associates who can help.

We're the broker-dealer and investment advisor for AXA Financial, Inc., a member of the global AXA Group, a worldwide leader in financial protection and wealth management. Our U.S. operating companies include Alliance Capital Management and The Equitable Life Assurance Society of the United States.

Professional Opportunities
What does it take to join our team? Find out here.

AXA FINANCIAL GOT THE INFRASTRUCTURE RIGHT

AXA Financial, Inc., a leading financial services company and member of the global AXA Group, wants to gain competitive advantage through superior customer service. To help make that goal a reality, they selected IBM's **WebSphere** and Candle Corporation's PathWAI™ suite of solutions.

Don Buskard's perspective:

"We use WebSphere to streamline and accelerate integration of key resources and applications held in different systems. Candle's PathWAI solutions are essential to implementing our WebSphere platform across the application infrastructure life cycle.

"**PathWAI Development** for **WebSphere** allows our application developers to work faster, avoid mistakes and reuse existing code. **PathWAI Monitor** for **WebSphere MQ** assesses the performance of our messaging middleware. And **PathWAI Dashboard** for **WebSphere MQ** provides an end-to-end view of critical applications that helps us map IT performance metrics to our business goals.

"A platform like WebSphere touches nearly all of your business-critical applications, so you need first-class products and services. That's why we selected Candle."

Don Buskard,
senior vice president and
chief technology officer
AXA Financial, Inc.

For the full story on how AXA Financial is using PathWAI solutions to deploy WebSphere with confidence, please visit **www.candle.com/pathwai** or call **(866) 488-4246.**

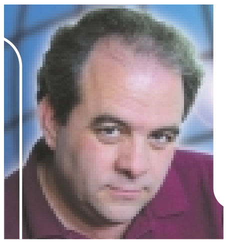
WebSphere ADVISOR GOLD
EDITORS' CHOICE 2003

e-Pro magazine APEX for WebSphere
2003 INDUSTRY DRIVER AWARD WINNER

Candle
Managing what matters most™

Outsourcing: Magic Bullet or Dirty Word? It All Depends on Your Perspective

BY JACK **MARTIN**



In the world of IT, outsourcing is either the dirtiest word you can utter or a brilliant one; it's all about who says it to whom and where it is said.

No matter who uses it, it is a word most often said in private. When corporate managers use the word, it is always mentioned in a most confidential fashion as a potential cost-cutting tactic, a magic bullet to increase margins.

When technical people use the word in public it is always with a hushed tone, as if speaking it aloud would give management the idea. In private it is discussed as if it were the greatest evil ever to befall the world, a faceless monster from far away.

The reality falls somewhere in the middle.

Outsourcing can be an extremely complex and complicated undertaking. Each piece of the process needs to be considered with great care and executed with precision. There is little margin for error halfway around the world. Once a company decides to outsource its code, programmers know their days are numbered. It's just a question of when the ax will fall. It is also just a matter of time before a major project goes completely out of control and craters, leaving hapless managers thrashing about with a project team in India.

So today we have corporate managers blindly sending work halfway around the world – and an endless drain of jobs overseas. Who came up with this latest corporate fad? How we got here is an interesting paradox.

Let's take a walk down memory lane. During the dotcom days, American code writers as a group became major prima donnas. It all started with the attitude, "I'm a programmer and I can wear anything I want to work," which was taken to the extreme by some people. Management was wearing suits, and in contrast the programmers looked like they came from some alien planet. The more outrageous the better.

From there, showing up at work at the same time as the rest of the staff became optional – the later the better – with the excuse that they were up all night writing code. It's true that a lot of code writers were up late at night writing code, but often not for their day job. An awful lot of people were busy writing code at night for dotcom business plans with IPO dollars in their dreams, while the more pragmatic moonlighted for other companies desperate for anyone who could write code.

Then the "I have to bring my dog to work" concept started. All of a sudden a menagerie of pets started showing up at work. Further, some programmers demanded and received trampolines. And not being happy even with all this, everyone was always ready to jump ship for more money and toys.

The final straw was the attitude, "I must work from home; you people are distracting me and I do much better work at home."

Well, to quote John Lennon, "The dream is over, what can I say. I was the Walrus and now I am John."

It is unfair to blame India for the loss of programming jobs. It was the prima donnas that got management going in this direction. Indians as a group are very polite and humble people who focus on doing a good job and are a pleasure to deal with. I haven't heard of any Indian companies demanding that corporate clients subsidize the cost of trampolines for their workers.

Next month I will discuss how to partially remedy this situation.

ABOUT THE AUTHOR... Jack Martin, editor-in-chief of ***WebSphere Journal***, is cofounder and CEO of Simplex Knowledge Company, an Internet software boutique specializing in WebSphere development. Simplex developed the first remote video transmission system designed specifically for childcare centers, which received worldwide media attention; and the world's first diagnostic-quality ultrasound broadcast system. Jack is coauthor of the upcoming book, ***Understanding WebSphere***, from Prentice Hall. **E-MAIL...** jack@sys-con.com

WebSphere® JOURNAL

ADVISORY BOARD
Richard Arone, David Caddis, Mike Curwen, Devi Gupta, Lloyd Hagemo, Barbara Johnson, Shannon Lynd, James Martin, Doug Stephen, Victor Valle

Editor-in-Chief	JACK MARTIN
Executive Editor	GAIL SCHULTZ
Contributing Editor	PATTI MARTIN
Product Review Editor	JAY JOHNSON
Special Projects Editor	RICHARD ARONE
Managing Editor	JEAN CASSIDY
Editor	NANCY VALENTINE
Associate Editors	JAMIE MATUSOW JENNIFER VAN WINCKEL

WRITERS IN THIS ISSUE
Marina Gil-Santamaria, Harvey Gunther, Lloyd Hagemo, Ravi Kalidindi, Adam Kolawa, Jim Liddle, Boris Lublinsky, Jack Martin, E. Kenneth Nwabueze, Jeffrey Payne, Adam Terrenzio

SUBSCRIPTIONS
For subscriptions and requests for bulk orders, please send your letters to Subscription Department.
SUBSCRIBE@SYS-CON.COM
Cover Price: $8.99/Issue Domestic: $149/YR (12 Issues)
Canada/Mexico: $169/YR Overseas: $179/YR
(U.S. Banks or Money Orders)
Back issues: $15 U.S. $20 All others

PRESIDENT AND CEO	FUAT KIRCAALI
PRESIDENT, SYS-CON EVENTS	GRISHA DAVIDA
GROUP PUBLISHER	JEREMY GEELAN
TECHNICAL DIRECTOR	ALAN WILLIAMSON
SENIOR VP, SALES & MARKETING	CARMEN GONZALEZ
VP, INFORMATION SYSTEMS	ROBERT DIAMOND
VP, SALES & MARKETING	MILES SILVERMAN
PRODUCTION CONSULTANT	JIM MORGAN
FINANCIAL ANALYST	JOAN LAROSE
ACCOUNTS RECEIVABLE	KERRI VON ACHEN
ACCOUNTS PAYABLE	BETTY WHITE
ADVERTISING DIRECTOR	ROBYN FORMA
DIRECTOR OF SALES AND MARKETING	MEGAN MUSSA
ADVERTISING SALES MANAGER	ALISA CATALANO
ASSOCIATE SALES MANAGERS	CARRIE GEBERT KRISTIN KUHNLE
CONFERENCE MANAGER	MICHAEL LYNCH
NATIONAL SALES MANAGER	SEAN RAMAN
ART DIRECTOR	ALEX BOTERO
ASSOCIATE ART DIRECTORS	LOUIS F. CUFFARI RICHARD SILVERBERG TAMI BEATTY
WEB DESIGNERS	STEPHEN KILMURRAY CHRISTOPHER CROCE
CONTENT EDITOR	LIN GOETZ
CIRCULATION SERVICE COORDINATORS	NIKI PANAGOPOULOS SHELIA DICKERSON EDNA EARLE RUSSELL
CUSTOMER RELATIONS MANAGER	RACHEL MCGOURAN

EDITORIAL OFFICES
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
WebSphere® Journal (ISSN# 1535-6914) is published monthly (12 times a year).
Postmaster send address changes to:
WebSphere Journal, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

WORLDWIDE NEWSSTAND DISTRIBUTION
CURTIS CIRCULATION COMPANY, NEW MILFORD, NJ

FOR LIST RENTAL INFORMATION:
KEVIN COLLOPY: 845 731-2684, KEVIN.COLLOPY@EDITHROMAN.COM
FRANK CIPOLLA: 845 731-3832, FRANK.CIPOLLA@EPOSTDIRECT.COM

Two years without a vacation.
The application's up. It's down.
It's up. It's down.

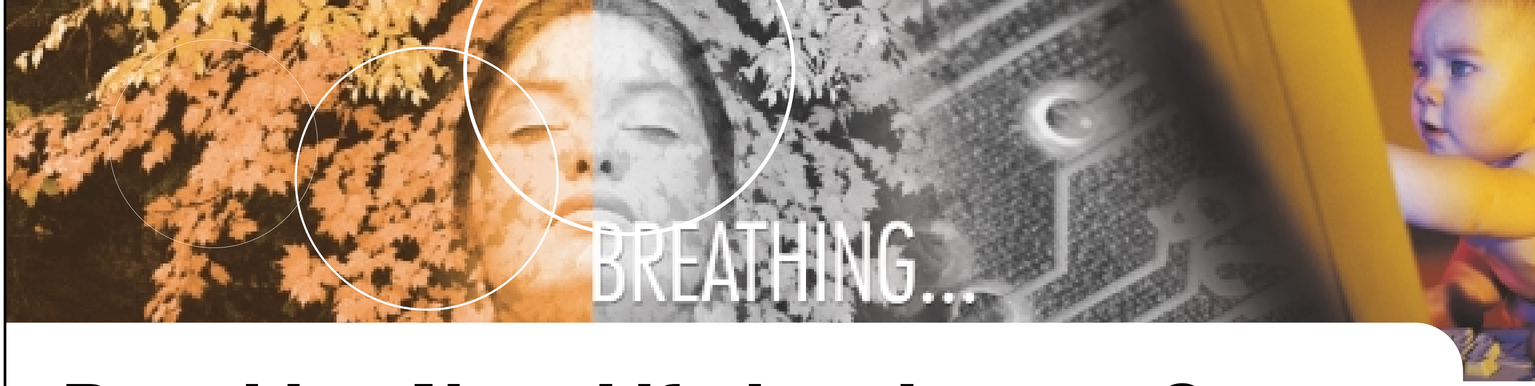
I'm to blame. Steve's to blame.
Someone's always to blame.

Not any more.

Get Wily.™

Enterprise Java
Application Management
1 888 GET WILY
www.wilytech.com

wily technology



Breathing New Life into Legacy Systems

Web services as an API

– BY JEFFREY PAYNE –

The technology world is abuzz with talk of Web services. Code warriors and suits alike are touting it as the next big thing. The incorruptible Apache Software Foundation has spawned a whole top-level project dedicated to it. Yet, I'd venture a guess that relatively few of us completely understand the concept – and even fewer sense an immediate need to dive right in.

Perhaps this is because Web services is pitched in fuzzy marketing language (from which it's difficult to extract anything concrete), most of which focuses on applying Web service technology in the consumer market and in B2B transactions. These are real applications and will likely become ubiquitous in the years ahead, but the immediate utility of Web services lies elsewhere.

The most exciting concrete thing about Web services today is its potential to finally deliver when it comes to integrating internal systems. If you could be a fly on the cube wall at most big corporations, you'd hear daily grousing about different systems not talking to each other and the need to enter the same data into several different applications, which costs us both in time and in the increased potential for data entry error.

It's an old problem, and most of us can recite a long list of acronyms we once thought would help us solve it: CORBA, RMI, COM, IOP, IIOP, RPC, and lately, EJBs. The problem with these methods of "distributing" applications is that they're proprietary to a specific programming language or a narrow subset of languages. EJBs work well if you're trying to interconnect a far-flung set of Java applications, but like it or not, much of the world's economy is transacted on big iron via COBOL and RPG (Report Program Generator) – and they don't speak bytecode.

All of them, however, speak text. For all the fanfare, Web services consists of little more than the exchange of text messages between peers. The text happens to be XML – and specially formatted XML at that – but text is still text. This inherent language independence – coupled with the fact that Sun, Microsoft, and IBM have actually agreed upon and contributed to the standards that constitute Web services – is highly encouraging. All this has given rise to the latest acronym being bandied about among the IT brass: EAI, or Enterprise Application Integration.

ABOUT THE AUTHOR
Jeffrey Payne is a Sun Certified Java Programmer and founder of Media Fortress, a Web service and J2EE consultancy based in Kansas City.

E-MAIL
jpayne@ mediafortress. com

How Web Services Actually Work

Web services, much like J2EE, is an umbrella term under which a number of related standards reside. Web services is often thought of as a layered stack of protocols consisting primarily of HTTP, XML, SOAP, WSDL, and UDDI. Other standards for things like authentication and two–phase commit transactions are in the hopper, but have yet to gain enough momentum for us to know if they will stick.

As I alluded to earlier, a Web service is basically a formatted exchange of XML messages, theoretically over any transport protocol, but HTTP is used often enough to make it the de facto standard. A typical Web service request consists of an XML-formatted message being posted to an HTTP server that responds with an XML-formatted reply. If we consider a simple state capital Web service, the raw HTTP exchange would look like Listings 1 and 2.

The flurry of tags and namespace declarations that surround the actual data in our state capital example is called SOAP. Although the acronym stands for Simple Object Access Protocol, SOAP is really a messaging protocol. It uses the concept of envelopes to wrap the XML data with the headers necessary to route, secure, and control the messages. In addition to pure message-based SOAP, a Remote Procedure Call model is specified by the SOAP 1.2 Adjuncts document as a standard extension to SOAP and is included in most SOAP implementations. While this is little more than a standardized way of encoding method calls as SOAP messages, it makes it possible to easily convert existing function or method libraries into Web services.

The whole idea behind Web services is to create or wrap some useful functionality using language-inde-

pendent SOAP messages. Doing this does little good unless there is a standardized way of describing the interface thus exposed by a Web service. WSDL, or Web Services Description Language, is that standard. WSDL provides information about the location of a Web service, its URI or identifier, along with what methods or "operations" are supported and the format of their associated parameters. It isn't absolutely necessary that you understand the details of WSDL to build Web services, as those flavors of WebSphere Studio that include Web service tools will generate the WSDL for you.

The last Web services standard we'll consider is UDDI or Universal Description, Discovery, and Integration. UDDI is a standard for Web service directories and at its most basic level provides a mechanism to store, find, and retrieve WSDL documents (and others) for public services. It allows Web services to be located by company name, service type, category, etc.

Viewed in the context of internal integration projects, a private UDDI server on your intranet can provide a central place for various project teams to publish the fact that they exist, along with their associated APIs. Project teams that must integrate their work with the work of other teams don't have to hunt for API documentation that may or may not exist. They just look up the other project's WSDL with UDDI and use a tool like WSAD to generate a Java client for it. It provides a way around the phone tag and meetings usually required to extract documentation from other project teams. The format of API documentation is consistent across all projects and can even be fed directly into code generators.

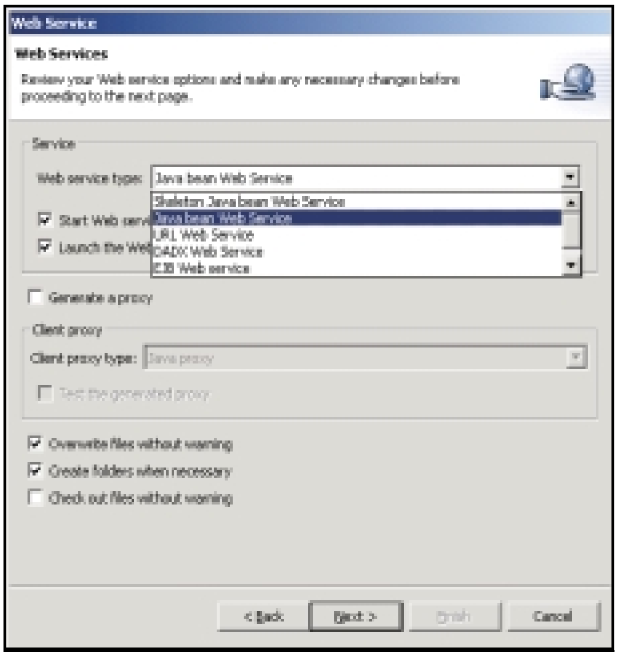
With the preliminaries out of the way, let's consider a simple example that makes use of all these acronyms and WSAD to generate a complete Web service. We'll stick to pure Java for now and delve into how legacy systems fit in later.

A Quick Tutorial

WebSphere Studio generates all of the plumbing required to publish a Web service for us. All we really need to do is create a simple JavaBean that performs a given function and let WSAD do the rest. Start by creating a new Web project called wsdj-example-web and a source package called com.wsdj.ws.example.

Next, create a new class called StateCapitalService and paste in the code in Listing 3. Note that there's nothing in this code that references SOAP or anything Web service–specific for that matter. This means that the code used to implement a service doesn't have to know it's a Web service. It's a JavaBean just like any other. This makes it extremely easy to convert old Java libraries (or Java wrappers for RPG and COBOL) into Web services without doing much extra legwork.

Before we actually create the service we need to create a local UDDI Registry in which to publish it. Do this by



Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

Service

Web service type: Java bean Web Service
- Skeleton Java bean Web Service
- Java bean Web Service
- URL Web Service
- DADX Web Service
- EJB Web service

[✓] Start Web serv
[✓] Launch the Web

[ ] Generate a proxy

Client proxy

Client proxy type: Java proxy

[ ] Test the generated proxy

[✓] Overwrite files without warning
[✓] Create folders when necessary
[ ] Check out files without warning

FIG. 1: SELECTING WEB SERVICE TYPE

STEP	WIZARD PANE NAME	ACTION
1	Web Services	Select "Java bean Web Service" and accept the default values as shown in Figure 1.
2	Web Service Deployment Settings	Accept the defaults.
3	Web Service Java Bean Selection	Use the "Browse Classes" button to select the StateCapitalService class you created earlier.
4	Web Service Java Bean Identity	Change the default value for Web Service URI to urn:StateCapitalService.
5	Web Service Java Bean Methods	Accept the defaults.
6	Web Service Binding Proxy Configuration	Accept the defaults. Generate proxy should NOT be selected because we'll be generating a proxy later using the UDDI registry.
7	Web Service Test	Accept the defaults.
8	Web Service Publication	Make sure the checkbox for publishing to the Unit Test UDDI registry is checked and the other checkbox is NOT checked.

TABLE 1: NAVIGATING THROUGH THE WEB SERVICE WIZARD

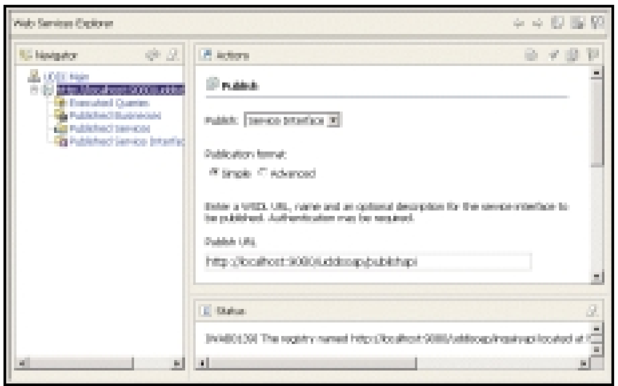
selecting "New" from the File menu. When the dialog comes up, select Web Services and Unit Test UDDI. Accept the default Registry Type (Cloudscape) and click "Next". Select the server you would like the local registry to run in and click "Finish". WSAD will then create the necessary resources, start the server, and load the Web Services Explorer.

Finally, we come to the point of turning a mere JavaBean into a full-fledged Web service. Select the wsdj-example-web project, open the context menu, and click "New". Select the Web Services category and Web Service as the wizard type. Click "Next" and use Table 1 to navigate through the subsequent wizard.

Provided you encountered no errors in the wizard, you have officially created a Web service. The next step is to publish it in the local UDDI registry. The wizard seems to suggest that it did this for you, but it didn't. It merely fires up the Web Services Explorer for you. Luckily, publishing is pretty easy. Select your local UDDI node from the Navigator pane (see Figure 2) and click the "Publish" button to the right of the "Actions" heading. Select Service Interface as the type of information to publish. Scrolling down, enter "admin" for the User ID and click the browse link next to the WSDL URL. In the pop-up window, select the wsdj-example-web project. The only WSDL file that should appear in the list will be the one for the StateCapitalService we just created. Click "GO" and finish up by entering a name and description for the service. Click "GO" on the publish page.

The WSDL document describing the state capital service has now been entered into the UDDI directory. At this point we can use yet another WebSphere Studio wizard to create a client for the service. Start by creating an empty Web project for the client called wsdj-example-client. Next, get the WSDL URL ready by copying it onto the clipboard before returning to the File menu and clicking "New". Select the Web Services category and Web Service Client. Accept the default values for the first wizard panel and click "Next". Paste the WSDL URL into the URL field and click "Next". Make sure the client project you just created is selected, step through, and finish the wizard.

If you take a look at the client project, you'll notice that a class called StateCapitalServiceProxy has been created with a getCapital() method signature identical to the one in the original service plus the getEndPoint() and setEndPoint() methods. To use the proxies, you must first set the endpoint value equal to the URL of the



Publish: Service Interface

Publication format: (●) Simple ( ) Advanced

Enter a WSDL URL, name and an optional description for the service interface to be published. Authentication may be required.

Publish URL: http://localhost:9080/uddisoap/publishapi

IWAB0136I The registry named http://localhost:9080/uddisoap/inquiryapi located at

FIG. 2: SELECT YOUR LOCAL UDDI NODE

SOAP RPC router, which is the servlet through which all RPC services are executed. This URL typically takes the form:

```
http://<hostname>:<port>/<application-
context>/servlet/rpcrouter
```

Admittedly, working our way through wizards panel by panel is a bit tedious, and the pathway described above should by no means be considered the only way to do it. The important thing to note about the preceding process is how little code was actually written. We essentially pointed the Web Service wizard at a JavaBean and let it work its magic. It generated the WSDL and deployed the service for us. We published the WSDL for our service in a UDDI registry, making it available for potential users to find. We then assumed the role of a client and used the published WSDL to generate a Java proxy for the service. We may have examined the two processes at a low level of detail, but with a little practice, building Web services and their clients becomes a quick, intuitive process.

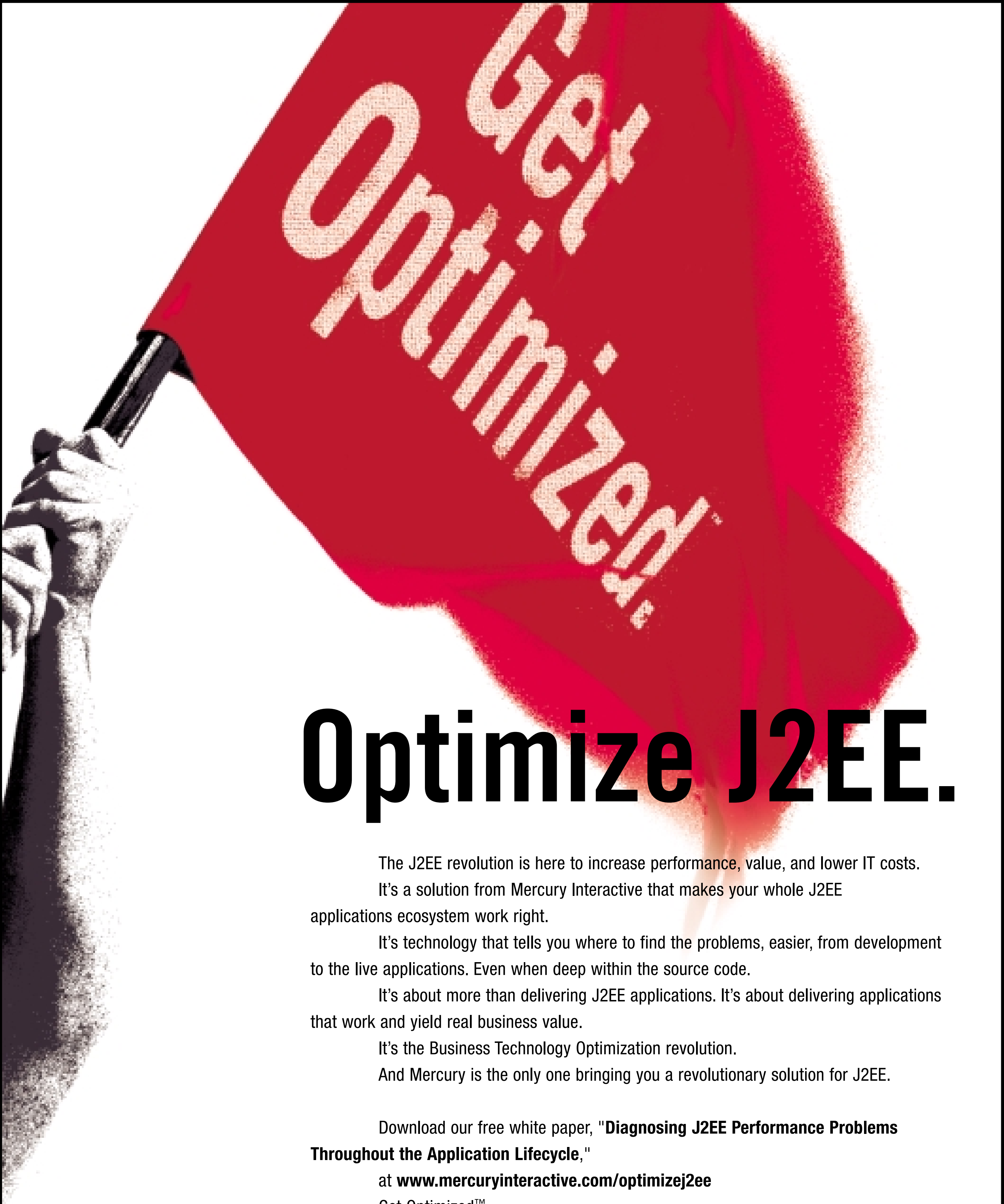
SOAP for Big Iron

COBOL and RPG may be losing favor, but they're far from dead languages. If making these apps accessible as Web services means rewriting them in Java or (shudder) C#, few prudent IT execs would blithely take such a risk. The good news is that several techniques exist for adding SOAP functionality to legacy systems without throwing the proverbial baby out with the bath water. For mainframes, IBM has a SupportPac module specifically for this purpose called SOAP for CICS. iSeries programmers have to go through WebSphere, which means wrapping everything in Java, but this isn't the Herculean labor it portends to be either. If you have WebSphere Development Studio for iSeries, you can generate Java wrappers for all of your RPG programs using fairly intuitive GUI tools.

The particulars of wrapping COBOL and RPG code with Java are arguably out of scope here, but knowing how you actually go about turning your legacy apps into Web services isn't nearly as important as knowing that you can. Building SOAP interfaces around legacy apps allows you to do a number of neat things.

First, taking an API approach and putting everything needed to interact with the app into the SOAP interface allows new user interfaces to be written in whatever language makes sense for that user interface. You could write a new Windows client using Visual Studio .NET or a new Web interface with WebSphere Studio. The demise of green screens means more intuitive user interfaces that dramatically lower training and help desk–related costs. Going the Web route even eliminates the need to support the software on user desktops. Client deployment consists of dropping a new EAR file into your friendly neighborhood app server and adding a new link to your intranet portal.

Second, apps that are merely walled in with SOAP interfaces don't have to be scrapped and rewritten. This obviously saves a lot of money by extending the life of applications, as well as mitigating the untold risks associated with tossing out perfectly good code whose only crime is not being Java.



Optimize J2EE.

The J2EE revolution is here to increase performance, value, and lower IT costs.

It's a solution from Mercury Interactive that makes your whole J2EE applications ecosystem work right.

It's technology that tells you where to find the problems, easier, from development to the live applications. Even when deep within the source code.

It's about more than delivering J2EE applications. It's about delivering applications that work and yield real business value.

It's the Business Technology Optimization revolution.

And Mercury is the only one bringing you a revolutionary solution for J2EE.

Download our free white paper, "**Diagnosing J2EE Performance Problems Throughout the Application Lifecycle**,"

at **www.mercuryinteractive.com/optimizej2ee**

Get Optimized.™

MERCURY INTERACTIVE

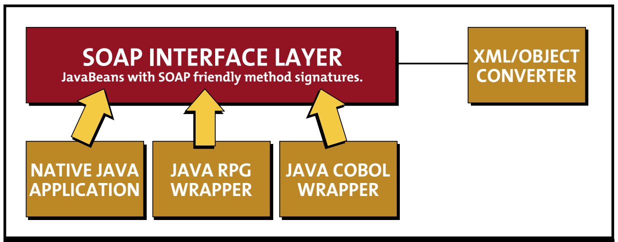


FIG. 3: GENERALIZED SOAP INTERFACE ARCHITECTURE

In addition, multiple applications with published SOAP APIs lend themselves to the development of single integrated clients that combine the features of each without forcing users to bounce around between systems.

SOAP APIs can deliver the holy grail of code reuse and component-based software development that actually works off the white board. Ostensibly, the main reason SOAP can actually make component-based software development possible is its indifference to programming language. And while that is a huge plus, I would argue that the UDDI and WSDL standards for publishing APIs are an even bigger plus when it comes to reuse. The reason is political. The more people that have to actively collaborate in order to share code, the less likely it is to actually happen. For a variety of reasons, be they inconsistent documentation formats, competitiveness between departments, or merger and acquisition–related political struggles, a centralized UDDI server isn't open to human interference. If the team developing an integrated client needs API docs, they know where to get them. No phone calls. No rooting around in shared folders for Word documents. No stonewalling. (A free open source UDDI server called Roundhouse that displays WSDL in a Javadoc-like format is available from www.mediafortress.com.)

Finally, a SOAP API makes it possible to convert the inner workings of legacy apps gradually. Once a SOAP API is deployed and users are weaned off proprietary clients, the internals of such applications can be slowly converted to a language such as Java without disrupting use or functionality of the system. An RPG application gets bits of Java introduced into it in the process of ongoing maintenance, and before long the hybrid app is more Java than RPG. Eventually the last line of RPG is replaced, and now you have a portable application that comes with no vendor lock-in strings attached.

The Catch

If you're salivating over the prospect of instantly modernizing the interfaces to your legacy apps by having your people spend a few weeks stepping through wizards, don't start popping corks just yet. Unless you take extra care to preserve the language neutrality and interoperability that makes us love SOAP so much, you could end up working yourself into a tight corner. The main gotcha involved with SOAP-enabling older applications revolves around complex data types such as objects and structures. Since such data structures are anything but language independent, any method or function being exposed via SOAP that makes use of such objects needs to be wrapped by another method that converts these objects into a language-neutral expression of structured data, which just happens to be something XML is awfully good at.

The SOAP specification itself does allow for the declaration of custom data types, but it's usually better to convert complex parameters into XML elements yourself than to let SOAP do it for you. Otherwise you get into the realm of creating custom object serializers and deserializers whose operation is closely tied to a specific SOAP implementation, which affects the portability of your own code. Clients that eventually make use of the interface will face a similar dilemma. Yet detaching the XML conversion of complex data types from the underlying SOAP provider doesn't alleviate the need to do it.

Assuming the bridge between old and new is made with Java, what emerges is a layer of JavaBeans specifically designed to have SOAP-friendly method signatures (see Figure 3), which can then be fed into Web service wizards. The most labor-intensive portion of any RPC SOAP interface, the XML/object converter, powers this layer. The form this converter will take is largely a function of the type of data being converted and the preferences of the developers who design it. It may be implemented by extending the XML transformer facilities of Xalan or some customized solution. Either way, every SOAP interface layer needs a black box into which Java objects go in and XML elements come out.

Building this black box is the trickiest part of building any SOAP API and requires the development of a standard XML representation for every object used by the application. Typically, this representation will ultimately manifest itself as an XML Schema file that can be used to validate the data coming into and out of the interface.

Another trouble spot with SOAP is security. Web services can be secure, but as of right now authentication and access control aren't part of the standards generally accepted to make up Web services. SOAP as implemented by Apache – and thus IBM – does support the idea of sessions, so session-based authentication can be used with SOAP just the way we use it with traditional Web applications.

You could always roll the dice with one of the competing security standards and hope you pick the one that eventually catches on. Or you could invent your own mechanism for authenticating users. At this point, there is no one perfect solution and certainly none that are part of implementations that aren't vendor specific. Your best bet would be finding a way to play nicely with whatever single sign-on solution your company is moving toward. Just be aware going in that dealing with security will be up to you.

The Road Ahead

Most of us have heard it said that one day in the not-too-distant future just about every business application will be J2EE or .NET. Recent studies by Gartner and others bear this out. For our sake, let's hope things are slanted a little more toward J2EE. However the pie chart eventually gets sliced, Web services and specifically SOAP promise to be a bridge between the two – and a viable way to build new applications from pieces of old ones. Web service applications will also emerge for consumer and for inter-business transactions, but more standards work is needed before Web services truly replace the phone and fax machine.

BREAK AWAY
FROM THE PACK

kenetiks
WORLDWIDE JAVA TRAINING & MENTORING

www.kenetiks.com
888.KENETIKS

- WEBSPHERE TRAINING
- PROJECT MENTORING
- COURSE LEASING

From development to deployment, Kenetiks shows you how to build J2EE applications for WebSphere Application Server and WebSphere Portal Server using WebSphere Studio Application Developer.

IBM Business Partner

An IT executive can't control what other IT executives do, or whether or not all the world's corporate transaction systems talk to each other. The Web service gurus aren't quite ready to tackle that one either. What Web services can do *now* is put language- and vendor-neutral APIs on top of any application, as long as you're willing to invest the time it takes to convert the object vocabulary of these applications into XML and back again. Once deployed, these APIs can be used to replace older clients, create integrated clients, accelerate the development of new applications by reusing pieces of the old ones, and shroud the innards of legacy apps under renovation.

Like all things in technology, these benefits don't come without trade-offs. Building XML Schemas for big applications with lots of complex data structures is no small task, and you'll have to find your own way when it comes to security. Yet all things considered, these annoyances are a small price to pay for the flexibility and long-term cost savings a Web service–based and fully integrated application infrastructure provides. And while portals as they typically exist today give companies the illusion of an integrated universe of applications, Web services can actually integrate them.

LISTING 1: HTTP POST REQUEST

```
POST /wsdj-example-web/servlet/rpcrouter HTTP/1.0
Host: localhost:8181
Content-Type: text/xml; charset=utf-8
Content-Length: 465
SOAPAction: ""

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:getCapital xmlns:ns1="urn:StateCapitalService" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encod-
ing/">
<stateCode xsi:type="xsd:string">KS</stateCode>
</ns1:getCapital>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

LISTING 2: HTTP RESPONSE REQUEST

```
HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Set-Cookie: JSESSIONID=0000FJH53QIGJEZCXK31F1UQASI:-1;Path=/
Cache-Control: no-cache="set-cookie,set-cookie2"
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Content-Type: text/xml; charset=utf-8
Content-Length: 481
Content-Language: en-US
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<ns1:getCapitalResponse xmlns:ns1="urn:StateCapitalService"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/
soap/encoding/">
<return xsi:type="xsd:string">Topeka</return>
</ns1:getCapitalResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

LISTING 3: STATECAPITALSERVICE CLASS

```
package com.wsjd.ws.example;
import java.util.HashMap;
import java.util.Map;

public class StateCapitalService {

   private Map capitalMap = null;

   public StateCapitalService() {
          super();
          init();
   }

   \public String getCapital(String stateCode) {
          return (String)capitalMap.get(stateCode);


   }

   private void init() {
          capitalMap = new HashMap();
          capitalMap.put("AL", "Montgomery");
          capitalMap.put("AK", "Juneau");
          capitalMap.put("AZ", "Pheonix");
          capitalMap.put("AR", "Little Rock");
          capitalMap.put("CA", "Sacramento");
          capitalMap.put("CO", "Denver");
          capitalMap.put("CT", "Hartford");
          capitalMap.put("DE", "Dover");
          capitalMap.put("FL", "Tallahassee");
          capitalMap.put("GA", "Atlanta");
          capitalMap.put("HI", "Honolulu");
          capitalMap.put("ID", "Boise");
          capitalMap.put("IL", "Springfield");
          capitalMap.put("IN", "Indianapolis");
          capitalMap.put("IA", "Des Moines");
          capitalMap.put("KS", "Topeka");
          capitalMap.put("KY", "Frankfort");
          capitalMap.put("LA", "Baton Rouge");
          capitalMap.put("ME", "Augusta");
          capitalMap.put("MD", "Annapolis");
          capitalMap.put("MA", "Boston");
          capitalMap.put("MI", "Lansing");
          capitalMap.put("MN", "Saint Paul");
          capitalMap.put("MS", "Jackson");
          capitalMap.put("MO", "Jefferson City");
          capitalMap.put("MT", "Helena");
          capitalMap.put("NE", "Lincoln");
          capitalMap.put("NV", "Carson City");
          capitalMap.put("NH", "Concord");
          capitalMap.put("NJ", "Trenton");
          capitalMap.put("NM", "Santa Fe");
          capitalMap.put("NY", "Albany");
          capitalMap.put("NC", "Raleigh");
          capitalMap.put("ND", "Bismarck");
          capitalMap.put("OH", "Columbus");
          capitalMap.put("OK", "Oklahoma City");
          capitalMap.put("OR", "Salem");
          capitalMap.put("PA", "Harrisburg");
          capitalMap.put("RI", "Providence");
          capitalMap.put("SC", "Columbia");
          capitalMap.put("SD", "Pierre");
          capitalMap.put("TN", "Nashville");
          capitalMap.put("TX", "Austin");
          capitalMap.put("UT", "Salt Lake City");
          capitalMap.put("VT", "Montpelier");
          capitalMap.put("VA", "Richmond");
          capitalMap.put("WA", "Olympia");
          capitalMap.put("WV", "Charleston");
          capitalMap.put("WI", "Madison");
          capitalMap.put("WY", "Cheyenne");
   }

}
```

TODAY'S MISSION: INTEGRATION — FULL SPEED AHEAD

KEY PRACTICE SKILLS:

WEBSPHERE APPLICATION SERVER

WEBSPHERE PORTAL SERVER

WEBSPHERE MQ

MQSERIES WORKFLOW

WEBSPHERE MQ INTEGRATOR

WEBSPHERE BUSINESS INTEGRATOR & CROSSWORLDS

WEBSPHERE COMMERCE SUITE

WEBSPHERE STUDIO APPLICATION DEVELOPER

VISUALAGE FOR JAVA

WEB SERVICES

DB2 & INFORMIX

PLANNING YOUR WEBSPHERE, WEB SERVICES OR PORTAL STRATEGY?

UPGRADING YOUR IT INFRASTRUCTURE?

NEED A JUMP START ON WEBSPHERE 5.0 AND WEBSPHERE PORTAL 4.1?

Prolifics' Jump Start Solutions will...

Educate. As 1 of 5 WebSphere service providers retained by IBM, Prolifics' Consultants are top-ranked by IBM in delivering WebSphere Solutions. We give your team an edge by transferring our extensive knowledge and experience.

Enable. We prepare your e-infrastructure for ease of application integration, migration and deployment. We recognize risks and delays that could hinder your project timeline and budget.

Empower. Providing your team with best practices, documentation and guidelines, we identify the best strategy leading to your must-meet business goals.

PROLIFICS' JUMP START SERVICES MENU:

- **Jump Start:** Extend your enterprise with WebSphere Portal 4.1
- **Jump Start:** Build, publish, manage, optimize with WebSphere 5.0
- **Jump Start:** Get started with Site Analyzer
- **Jump Start:** Integrate WebSphere MQ into a J2EE environment
- **Jump Start:** Web services – build to integrate

Call 800.675.5419 or 212.267.7722

http://www.prolifics.com

Prolifics

IBM Business Partner

Solving Business Problems with Technology Solutions since 1978
Expert Delivery of WebSphere, Integration and Business Transformation Solutions

For a FREE white paper on how to Integrate Enterprise Resources Using Web Services Technologies
Go to: http://www.prolifics.com/webservices

PRIORITY CODE: DJ0702JS

e-business

Front Controller pattern simplifies integration challenges

One Access Point to Rule Them All

BY LLOYD **HAGEMO**
AND RAVI **KALIDINDI**

The concept of a central point of access to an application or set of applications is not new. For more than 30 years, companies have been writing, enhancing, and maintaining applications written to transactional systems such as CICS and IMS for IBM OS/390 mainframes. These mature mainframe applications are similar to the J2EE applications developed today because they provide an environment for building customer transaction applications that support distributed enterprise computing.

ABOUT THE AUTHOR
Lloyd Hagemo is a senior director for Candle Corporation's Application Infrastructure Management Group. He is responsible for WebSphere tools development. Lloyd has led the successful development of more than 20 products for the WebSphere environment, including operating system utilities, network performance and tuning products, WebSphere MQ configuration and management tools, and application integration solutions.

E-MAIL
llyod_hagemo@candle.com

An effective application infrastructure must include a common point of access to perform application tasks such as authorization and routing. Centralized application management must also include the ability to deploy new functionality incrementally, fulfilling organizational requirements to continually enhance business processes.

IBM developed the first mainstream central point-of-access solution when it introduced the Network Logical Data Manager, which allowed systems programmers to access – from a single screen – multiple applications concurrently running in CICS, IMS, or Service Network Architecture (SNA)-based subsystems without logging on and off of each subsystem to retrieve information. Organizations quickly saw the advantages of creating a single view of applications that were previously siloed. For example, a bank teller would use this technology to provide faster, more comprehensive customer service by having a single view of a customer's checking account, savings account, certificates of deposit, and other account information.

The ability of programmers and developers to maintain centralized control is essential to the successful development, deployment, and management of J2EE applications. In J2EE environments, the Front Controller pattern provides a standardized single point of entry for processing Web requests. It acts as a router that centralizes functions – such as view selection, security, and templating – and applies these functions consistently across all pages or views.

This column outlines the different uses of the Front Controller pattern, describes how one company used this technique to its advantage, and offers alternatives for centralized control.

Advantages of the Front Controller Pattern

The Front Controller pattern represents a proven way to enable simultaneous access to multiple J2EE-supported applications. It delivers the following benefits:

- Provides a central access point for applications
- Creates an ideal location to initiate authentication and authorization
- Ensures code reuse and reduces the incidence of redundant code being used in several applications
- Provides common services required to process requests
- Separates individual roles within an application to improve testability
- Does not restrict the number of controllers used for a solution

Success Story

Web development offers the advantage of building a solution incrementally. The following real-world example outlines how a company began its Web development by creating an order entry system that enabled customers to purchase office supplies online. The company's customer service telephone number was available for customers to change, cancel, or check the shipping status of their orders. Internal call center personnel also had access to the shipping and order systems. The company soon expanded its Web-based system to provide invoicing, instead of requiring approved customers to pay by credit card for each order.

When Web site traffic increased, the company added a customer relationship management (CRM) system to expand customer service and reduce costs. The CRM system enabled customers to use the Web site to perform all tasks previously available only via the customer service telephone number. The company created the shipping and invoice systems by using simple URLs that pointed to servlets. The servlets performed database lookups using search criteria, such as customer name or order number (see Figure 1).

Front Controller Pattern Context

The company's four applications – CRM, order management, shipping, and invoicing – were integrated using common classes and customized interfaces. (A class provides a template for actions that make up a specific object or application.) The Web site architecture also required duplication of Java Database Connectivity code among the CRM, order management, and shipping applications.

The invoice system was used externally by customers through the CRM interface and internally by call center staff. The CRM and order systems used application-specific Front Controller patterns to provide routing within each application. The code used for customer authorization was duplicated in both systems. The reuse of classes, multiple interfaces, and duplication of SQL code created a severe maintenance problem and was affecting performance.

Company IT managers realized that the solution to the Web site maintenance and performance problems would require tuning application code and defining cleaner interfaces. The IT team initially researched portal technology to provide a tighter level of integration among the multiple Web applications. The portal solution was not viable, however, because the organization did not want to purchase the hardware and software required for a portal server. Further, the company required a solution that did not entail extensive training of its developers and programmers.

A more effective option, the company decided, was to design a Front Controller pattern to address existing Web site performance slowdowns and support additional applications in the future. The Front Controller pattern addressed the requirement for a single location to authorize internal and external users. IT created a user interface that provides a specific set of selections for users to access back-end applications, common helper functions for database access, and a command processor to support well-defined XML application interfaces.

Figure 2 illustrates Front Controller pattern deployment, which enabled the company to centralize key Web site application functions to improve performance and minimize system complexity.

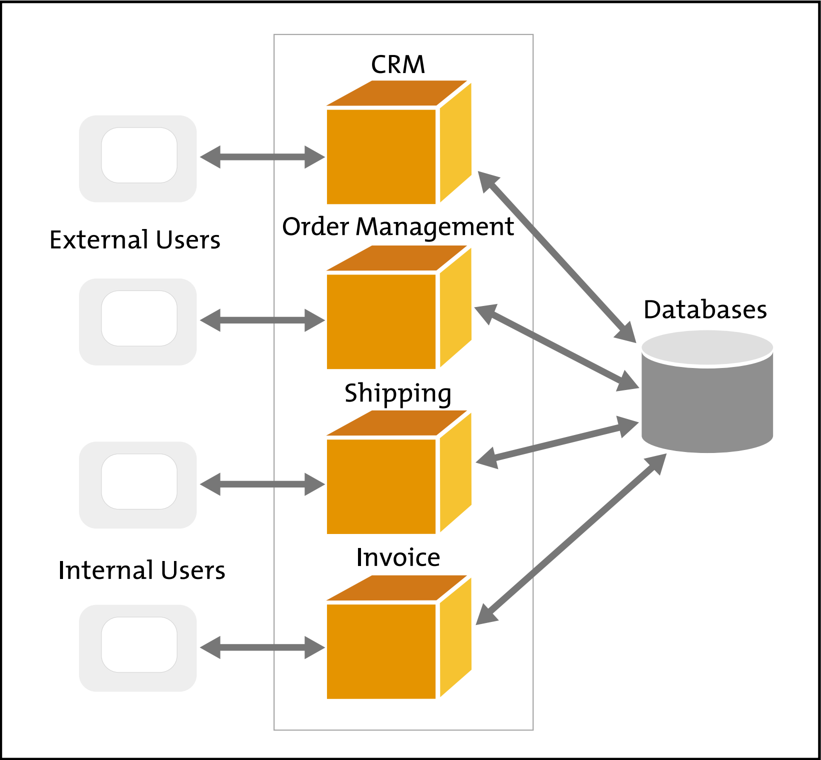


FIG. 1: HOW WEB SITE VISITORS INITIALLY ACCESSED EACH APPLICATION

A Front Controller pattern provides authorization and a customized user interface via helper beans. The application interfaces were standardized, enabling the CommandHelper class to be used to drive different application functions. The CommandHelper class eliminated the requirement for duplicate functions that had been included in each of the Web site applications. For example, the company performed the XML conversions in the Front Controller pattern by using the CommandHelper class to support the new and existing application interfaces. Additionally, the database tables control the Front Controller pattern definitions, which enable the Web-based system to scale quickly to support additional applications as the site expands.

Front Controller Pattern Implementation

The example outlined in this column illustrates how the Front Controller pattern acts like a hardware switch or a router that controls and routes clients' requests to different application servers. The Front Controller pattern is the point that controls all client requests. There can be different implementations for the Front Controller pattern, depending on the front end or the middle-tier pattern configurations. An application can, for example, use the CommandHelper or Dispatcher pattern to create a system that is modular and easier to maintain. Additional transactions (commands) can be easily added by changing the command factory. This allows additional functions and applications to be easily added, expanding the capabilities of the application.

As shown in Figure 3, the Front Controller pattern processes a request and executes the related command using the CommandHelper class. The CommandHelper class creates a command object for each client request. If the client requests an order placement, for example, the controller executes the OrderPlacementCommand class. The controller calls the execute() method

ABOUT THE AUTHOR

Ravi Kalidindi is a senior software engineer in Candle's Application Infrastructure Management Group. Ravi has worked with Java since its inception and has published several articles that focus on J2EE best practices.

E-MAIL

ravi_kalidindi@candle.com

of the OrderPlacementCommand class and receives the order confirmation JavaServer Page (JSP). Then the controller dispatches the control to the order confirmation JSP to display the results to the client.

Listing 1 shows the sample code for the OrderController class, which controls all the requests related to the order.

Multiple design patterns are available to address specific business and operational requirements. For example, the controller can communicate with a CommandHelper pattern to delegate requests to a related business component. Keywords within the URL can be used to trigger specific servlets or Enterprise JavaBeans to process a transaction. Although a single class can maintain all the controller logic, it is a good practice to create multiple functional controllers to manage specific tasks, such as OrderController, InventoryController, and ReportsController.

This design promotes easy maintenance with different functional controllers, and eliminates performance and maintenance problems associated with a single monolithic controller. It is a common practice today to implement an application using a controller. These applications (i.e., OrderController, InventoryController, ReportsController) provide specific routing and functions related to that application. To provide a solution, the Front Controller pattern can be used to tie them together.

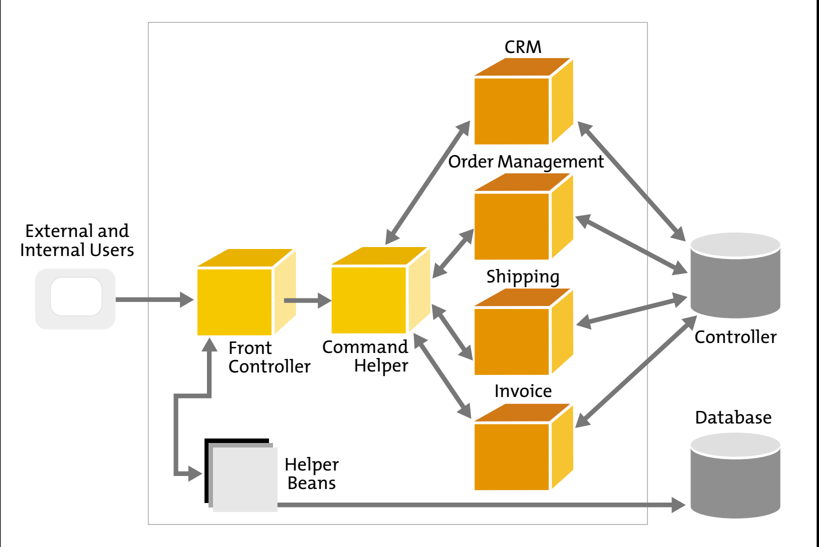


FIG. 2: FRONT CONTROLLER PATTERN DEPLOYMENT

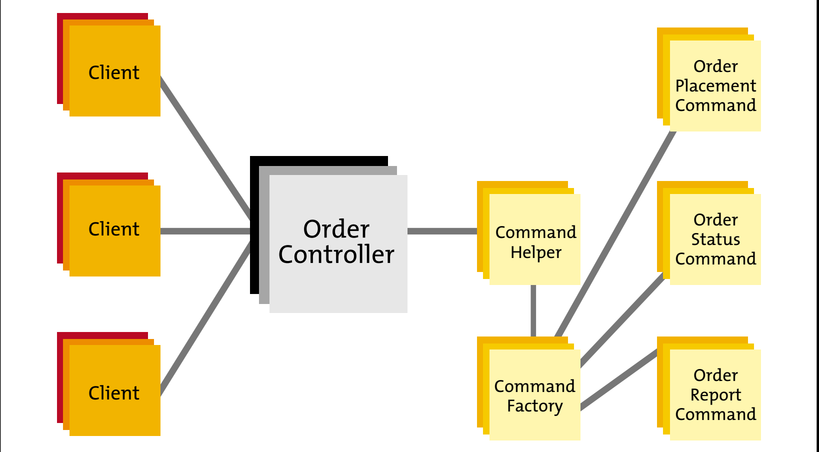


FIG. 3: HOW A FRONT CONTROLLER PATTERN "TALKS" TO MULTIPLE CLASS COMMANDS

Conclusion

The Front Controller pattern is ideal for integrating components and applications, and reducing the duplication of code by providing a common point of access to perform such tasks as authorization and routing. The Front Controller pattern allows organizations to centralize user definitions, enable access to authorized applications, integrate applications, optimize interfaces using XML, and reduce duplicate code. The pattern can be used for simple information retrieval and for extensive database-driven solutions that route information among multiple applications. To reduce complexity, it is an acceptable practice to have multiple controllers in a solution.

Time-to-market is a key business driver for application development. Using an extensible model allows application development organizations to release functionality as it becomes available. A well-architected system that leverages best-practice design patterns, such as the Front Controller pattern, makes this job easier.

LISTING 1

```
public class OrderController extends
HttpServlet implements Servlet {

  public void doGet(HttpServlet
  Request req,  HttpServletResponse
  resp)
     throws ServletException,
     IOException {
     try{
         Command com = Command
         Helper.getCommand(req);
         String viewPage = com.exe
         cute(req,resp);
         RequestDispatcher rd =
         getServletContext().get
         RequestDispatcher(viewPage);
         rd.forward(req,resp);
     }catch(Exception e){
         System.out.println
         (e.getMessage());
     }
  }
  public void doPost(HttpServlet
  Request req,
     HttpServletResponse resp)
     throws ServletException,
     IOException {
     doGet(req,resp);
  }
}
```

THE LEARNING CONTIUM
AD

Managing the WebSphere Platform

Getting an end-to-end view of your applications

– BY MARINA **GIL-SANTAMARIA** –

Your applications have just gone through a reasonably in-depth testing cycle, and now they are finally deployed in your WebSphere production environment. Great! So now I wonder... what about your WebSphere business processes? Do you understand your applications' topology, know which components participate on which flows, and in which system they "live"?

And what about the end-to-end picture? Do you know where your routers are, which WebSphere Application Server is connected to which Web server or to which WebSphere MQ system? And what about your connectivity with your back-end systems? Do you know which JavaBean is accessing which table, and where all the messaging systems are physically located? If your answer is no, this article will show you how you can better understand and manage your WebSphere applications.

From WebSphere Application Server to the WebSphere Platform

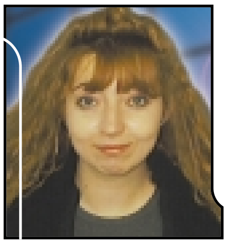
Before I discuss different approaches to managing the WebSphere platform, let's begin by defining what I mean by this term. WebSphere Application Servers (WASs) – and application servers in general – which for the most part once simply stored and served up content, are now part of bigger platforms, known as "application server platforms," that offer application hosting, integration, portal, data management, security, and other functions. In fact, IBM's WebSphere product line today contains more than 350 products. You can see a graphical representation of the main components of the IBM WebSphere 5.0 platform in Figure 1.

When you look at it, it seems complex. Maybe you are even thinking, "Hmmm, in my organization we are not using all of these." Don't be too sure. You might be right in the sense that you might not be using CICS, for example, but I am certain that you at least have Web servers at the front end (WebSphere relies on available Web servers to present content to the end user); databases, constantly collecting and supplying critical information; messaging systems; and network elements like routers or firewalls deployed across your organization. You might even have WebSphere MQ or legacy applications running in your environment as well.

A problem with any of these elements will impact your system's ability to present content to the end users accessing your WebSphere applications. So let's take a look at the WebSphere platform from a different angle, an enterprise implementation perspective, like the one portrayed in Figure 2.

As you can see in Figure 2, WebSphere applications expand beyond the WebSphere application server, and are deployed across extended underlying infrastructures on both distributed and mainframe environments. This means that even though your main focus is WebSphere, you shouldn't forget that Web transactions traverse a path of complex, interdependent entities, any of which are potential bottlenecks or failure points. Not only is every element of the chain responsible for presenting content to the end user, but problems can happen in the source code of a particular JavaBean, in a specific method's call to a back-end database, or in a router.

As infrastructures become more and more complex, applications too are becoming more and more complex. When you think about a typical J2EE production application you are going to see a few hundred components – JSPs, servlets, EJBs, methods, etc. – layered, interconnected, and highly distributed across different systems. In addition, business processes within the context of WebSphere are particularly dynamic in nature, as application cycles have shortened considerably, and applications are moved into production faster than ever before. At the same time, there are multiple departments involved in the process of building, deploying, and maintaining applications: Operations, WAS Administrators, DBAs, system administrators, Development, QA, network team, etc. So, how do you manage all of these?



ABOUT THE AUTHOR
Marina Gil-Santamaria is a product manager for Unicenter Management for Web and Application Servers. During the past five years with Computer Associates, Marina has held various positions, including senior systems engineer and specialist across different divisions of CA. Marina holds an MS in electrical engineering from the Universidad Politecnica de Madrid, Spain.

E-MAIL
marina.gilsantamaria@ca.com

Traditionally, application management was characterized by a horizontal approach, in the sense that a domain management group was responsible for individually maintaining each silo, e.g., DBAs were responsible for maintaining databases; WAS administrators were monitoring WAS systems; network administrators were overseeing routers and firewalls, etc. These domains were pulled together into frameworks and platforms, but the level of integration was minimal, i.e., leveraging a single console without meaningful integration of information and services.

This approach is no longer feasible, as it will not allow you to quickly locate and identify where a problem is really happening, and which team needs to address it. Let me give you an example. Let's assume that you are independently monitoring your DB2 systems, and you are suddenly alerted to problems with specific tables escalating locks and timeouts. Most of the updated SQL statements, if not all, are entity bean–driven, and the beans create their own SQL statements. Because you don't have a correlated WAS/DB2 management in place, you really don't know which specific application components are executing those SQL calls. What is the consequence? As you can only speculate which applications are causing the problem, you can only help by partitioning tables that are not partitioned, a temporary fix. Because the root cause of the problem hasn't really been found, it is very likely the problem will occur again.

Therefore, effective management of WAS environments requires the latitude for in-depth monitoring both across each silo and end to end, and with the capacity to correlate performance from the end-user experience down to the individual database, WAS, network element, or Java component that is really causing a performance bottleneck or degradation.

Divide-and-Conquer Technique

Although WAS comes with it's own embedded management tools, when defining your strategy for managing your applications built on top of WebSphere you should think about end-to-end management. This means managing from a single location all the different elements that form the WebSphere platform, monitoring application components and their dependencies on the underlying infrastructure, and being able to correlate performance metrics end to end to rapidly find the true root cause of problems or performance degradations in real time. I like to refer to this end-to-end management strategy as a "divide and conquer" approach, in the sense that you start by considering your application architecture in terms of individual

- Elements of the infrastructure
- Application components
- Topological relations, of both application component to application component, and application component to underlying infrastructure

Once you have management tools in place to discover and monitor the performance of individual infrastructure elements and application components – tools that allow you to visualize from a central location topological relationships as well – you can start building up your management and go to the next level, which is monitoring interactions among components and end-to-end SLAs (service-level agreements).

Managing Infrastructure and Application Components

Let's back up at this point, and take a look at the first step of this approach in more detail. You need individual management solutions to manage each piece of the infrastructure, e.g., WAS, WebSphere MQ, Web servers, databases, CICS, messaging systems, network elements, etc., – and to map different management solutions to all the elements of the WebSphere platform. At the same time, these individual management solutions, or agents, should have integration points among each other at different layers to be able to correlate performance metrics end to end. Some of the integration points that you should look for in your management tools are:

- ***Global discovery:*** Ability to discover the end-to-end WebSphere infrastructure and application components from a central location
- ***Central visualization:*** Helps you visualize problems, drill down to the affected area, and route issues to the department that needs to fix them.

"Web transactions themselves are complex and heterogeneous operations, as personalization and different interests will usually lead end users to traverse different paths on the same application"

- ***Single point of notification/alerting/root-cause analysis:*** Allows intelligent filtering out of events
- ***Customizable integrated reporting:*** Ability to group information in terms of individual elements, topological relationships, or business processes for both historical and real-time reports

Now that you have your agents in place, the next step should be automatically discovering your application components, where they have been deployed, and their dependencies down to the business logic layer. For example, a particular EJB method in my Checkout EJB is executing a SQL query on my DB2 system A. Once you have the topological information of your enterprise application in place, you should start by looking at the performance metrics of the different systems, as well as the performance of each individual application component. You should look for a management tool that is proactive in the sense that it is based on configurable thresholds so that when these thresholds have been reached, alerts will be sent out to the central console, and automatic corrective actions can be taken to fix the problem without IT intervention.

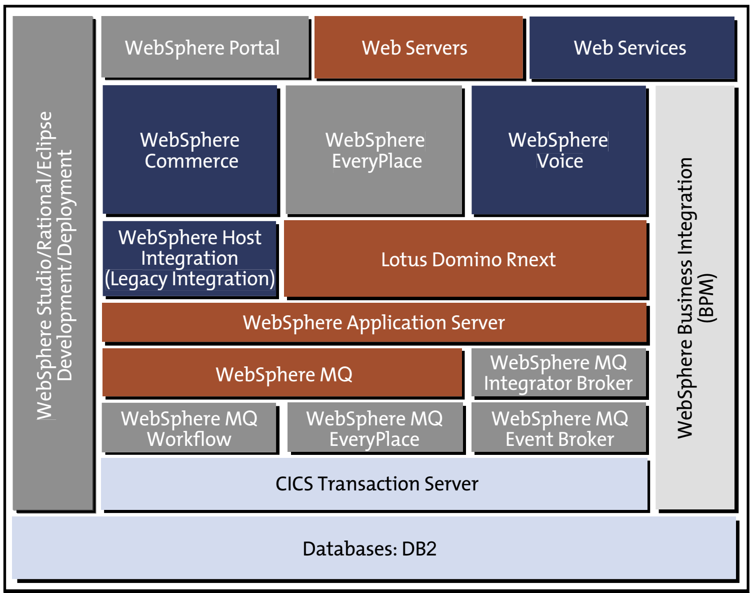

WebSphere Studio/Rational/Eclipse Development/Deployment
WebSphere Portal
Web Servers
Web Services
WebSphere Commerce
WebSphere EveryPlace
WebSphere Voice
WebSphere Business Integration (BPM)
WebSphere Host Integration (Legacy Integration)
Lotus Domino Rnext
WebSphere Application Server
WebSphere MQ
WebSphere MQ Integrator Broker
WebSphere MQ Workflow
WebSphere MQ EveryPlace
WebSphere MQ Event Broker
CICS Transaction Server
Databases: DB2

FIG. 1: WEBSPHERE 5.0 PLATFORM

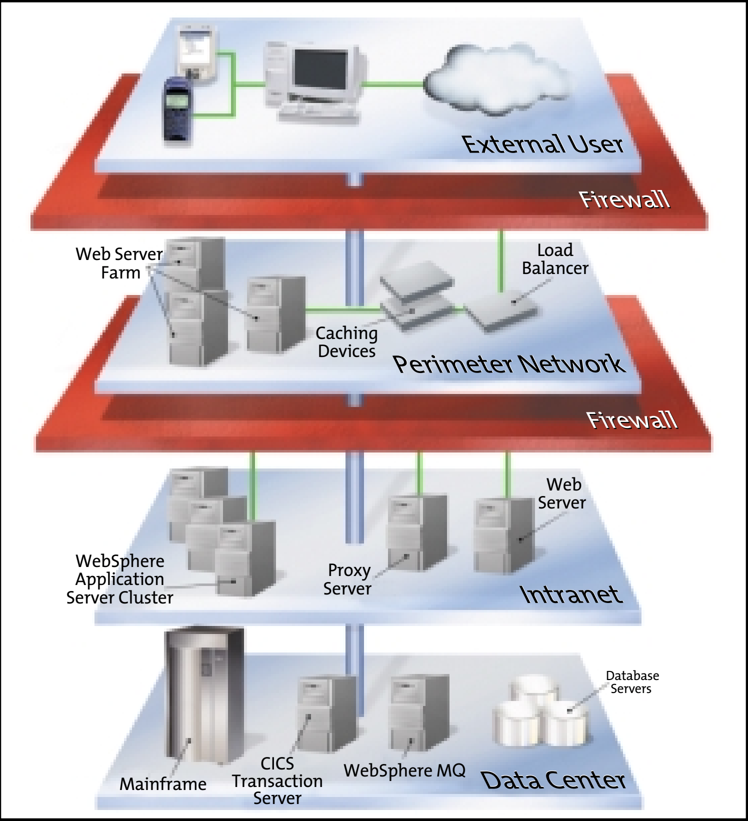


FIG. 2: EXAMPLE OF AN ENTERPRISE WEBSPHERE IMPLEMENTATION

Managing Applications in the Context of Flows

Now that you can oversee the availability, health, and performance of individual systems and each application component from a topological perspective, the next step should be managing interactions among application components in the context of flows. There are two ways to look at interactions, from a Web transaction perspective and from a WebSphere business process approach. Let's begin by focusing first on Web transactions.

Because of the complexity of the WebSphere platform, managing Web transactions as they flow across your extended environment is critical to ensure a positive end-user experience. Each time an end user accesses a Web application, multiple transactions take place: from the Web browser to the Web server, from the Web server to a WAS system, from a WAS system to the database, and so on. Web transactions themselves are complex and heterogeneous operations, as personalization and different interests will usually lead end users to traverse different paths on the same application.

Vendors have been addressing transaction management from two different perspectives: real-time versus synthetic monitoring. The reasoning behind these two approaches is that real-time monitoring can help you quickly detect and pinpoint application problems to an offending JavaBean method, servlet, or database connection in real time as soon as they happen, while synthetic monitoring proactively exercises applications to help prevent application crashes or wrong return data before real end users encounter those situations.

An ideal management solution should deliver both approaches. In fact, you should start by identifying your real business objectives or scenarios – e.g., my goal is to get users to register for a seminar, or to sell item A – to identify which Web transactions are critical in your environment. Each one of these critical business scenarios or transactions should be managed synthetically and in real time as an object. Also, if any of the application components is executing SQL queries, you should look for a management tool that captures those in real time as well, presenting performance information for both the component that executed the SQL query as well as the SQL query statement, to help you determine if a problem is on the application server side or on the database side.

Related to managing WebSphere business processes, a management tool can allow you to work with abstractions of the real processes within the WebSphere Application Server platform, and monitor as a unit all the elements across the end-to-end environment that are critical to your key business processes, including specific application components, infrastructure elements, Java Virtual Machines (JVMs), WebSphere MQ systems, Web servers, database tables, and routers. Because of the dynamic nature of WebSphere environments, a management tool should handle these abstractions as dynamic entities, and quickly update and reflect changes in application components, application dependencies, or underlying infrastructures. In addition, a management tool should offer rule-definition capacities so you can associate rules to govern each of your WebSphere business processes or abstractions. Hence, these rules will help you determine, in the event of an infrastructure element malfunctioning, which business processes are going to be impacted, so errors can be prioritized and addressed accordingly.

Managing End-to-End SLAS

The effectiveness of an application can be measured accurately only from the service consumer's viewpoint. Now that you are overseeing infrastructure, application components, transactions, and business processes, the next step is managing end-to-end service-level agreements to ensure a positive end-user experience across the enterprise. A management tool should allow you to build a higher view out of the underlying complexity of routers, switches, application components, JVMs, response times, and the like, correlating data from different sources across the enterprise, calculating availability percentages, and sending potential breaches of end-to-end SLAs to the central notification console. Specifically, it would be very useful if you could use the WebSphere business processes or abstraction units I discussed in the previous section as a data point for your SLA management tools, so you can measure effectiveness from an application-centric perspective as well. Another useful feature is the ability to customize and present SLM reports on a business's process, geography, organization, customer service goals, and business-hours basis, allowing IT to refocus on services that contribute the most to corporate profitability. It will also enhance other business goals such as market share or shareholder value.

Within Computer Associates' Unicenter portfolio, we map different Unicenter solutions to the different elements of the WebSphere platform so we can monitor the underlying extended WebSphere infrastructure, as well as the health, availability, and performance of individual application components and their interactions with each other in the form of Web transactions.

All of our Unicenter solutions use CA Common Services (CCS) – a common set of APIs that leverage Web services– enabled technologies – to facilitate integration and communication among them, and correlate performance metrics across silos, and vertically across business processes. CCS provides multiple points of integration at different layers from which you can manage your end-to-end infrastructure, while also providing a central point of notification with advanced root-cause analysis capabilities.

Conclusion

Whatever management tool you choose, the goal is to maintain your critical Web applications and exceed user service levels by proactively managing the performance of your WebSphere platform and its underlying J2EE technologies – including real-time transaction monitoring, and your customized business logic – to help you locate and resolve problems in today's extended WebSphere environments.

KNOW IT ALL

IBM-certified training in WebSphere MQ, Workflow and J2EE.

In today's IT environment, you need to be a know-it-all. Because what you don't know can hurt you – and your business. Let MQU help make you the master of your IT domain, with IBM-certified WebSphere training that teaches you how to tap the rich functionality of this powerful technology.

Enroll today. Visit www.mqsoftware.com or call 800.998.2858 for course descriptions and schedules.

MQSoftware™ University

IBM Business Partner

Knowledgeable instructors. Hands-on format. Classroom or onsite courses. Customizable curriculums.

A technical overview

Understanding Straight-Through Processing

BY JAMES **LIDDLE**

Straight-Through Processing (STP) is a term associated with workflow and business process management technologies. STP is the automation of a process flow, from invocation to execution. STP allows the seamless and accurate transactional exchange of information electronically. It can involve many different information stores and parties within the process flow, and it reduces the need for human intervention.

ABOUT THE AUTHOR

Jim Liddle, principal solutions architect for Versata, Inc., has worked within the IT industry for over 14 years, the past 4 working with J2EE technology. Jim's background is in the design of large-scale enterprise systems. He holds an MSc in data communications.

www.versata.com

E-MAIL

james_liddle@versata.com

STP is often associated with the financial services and securities industry, as this is where its use has been prevalent, due to a change in legislation (Rule 15c6-1) by the Securities and Exchange Commission. This shortened the settlement time for trades from 5 days (T+5) to 3 days (T+3). Previously it was possible to manually settle all trades, but the shortened cycle time, coupled with increasing volumes, meant that automation was the only real solution to T+3 trade settlement. This ultimately involves integrating automated processes such as order management, trade capture, confirmation, confirmation matching, settlement, general ledger accounting, cash and asset reconciliation, and transaction and account reporting.

Business processes and STP are inextricably linked; you won't often find reference to one without the other. This is because the first stage in the automation of a "business transaction" (not to be confused with ACID transactions, which a business transaction is composed of) is the building of a consistent platform for disparate processes that reside in different systems. BPM technology provides this consistency, residing above an organization's existing internal systems and applications.

STP is increasingly being used in other industries and applications where there is a need to automate flows and orchestrate services. This has resulted in a number of different approaches, patterns, terms, and technologies that I will explore in this article.

STP Implementations

Service-based architectures and Web services technologies have had a big impact on how and why organizations want to construct automated process flows, and also on the technology used to implement them.

Many organizations have utilized such architectures to build granular services that interact with their legacy back-office systems. The next step is to orchestrate these granular services to provide a higher-level business service. This can and has been done using code, but it can prove to be inflexible and difficult to change and maintain. Organizations are increasingly looking for more productive ways to orchestrate and automate flows, which provides them with greater flexibility and productivity, and often involves a visual design tool and/or code generation.

Traditionally business process technology has been associated with long-lived asynchronous-type flows that often involve human interaction and multiple ACID transactions. Process templates are defined at design time in a visual user interface and the process meta-information is persisted at design time using a markup language. An example of this is flow description markup language (FDML), which is used by MQSeries Workflow. Other notations that are used center around the process vendors' support for proposed standards, such as WS BPEL and BPML. An overview of these standards is outside the scope of this article but is offered in my earlier article, entitled "Process-Driven Architectures for WebSphere Application Server" [***WJ***, Vol. 2, issue 1].

When deployed, the process template is persisted to a database that the process engine uses to store state and process information. The state of the running process is persisted at each step by the process engine to ensure reliability and failover, and also as a mechanism to deal with long-running processes, which may take days, months, or in extreme cases, years.

Figure 1 shows this type of long-lived process, using as an example a bank loan application in which three different parties – the customer, the bank agent, and the insurance underwriter – interact over an infinite period of time. The completion of each step moves the process flow on to the next step and there may be rules and timers at each step.

This model does not work for STP flows, however, which are often synchronous and may need to be executed in real time. Process vendors such as Versata have tackled this by providing STP activities in which the whole flow is synchronous and can execute within the same transaction, while state is only persisted at the beginning and end of the process.

Figure 2 shows this type of STP process using an STP process aimed at preventing money-laundering as an example. In this example the synchronous STP process is invoked from a session bean, which submits a payload that is used to check client details, check against sanction lists (hosted as a Web service), run profile rules on the data submitted, log noncompliance, and add details to a user's worklist for further investigation, before returning a response as to success or failure.

This STP process has been constructed visually, interacts with prebuilt services, is synchronous, executes in a single thread and a single transaction, and is far quicker than a traditional BPM process due to the lack of persistent I/O at each step. This process can also be used as a subprocess within a higher-level asynchronous process, enabling the design of business flows that accurately reflect synchronous or asynchronous requirements.

The response for validation must occur within three seconds, according to the SLA with their customers. This is currently done using a legacy framework that the company built but which does not take advantage of industry-based middleware or modern programming languages. Consequently, it is becoming difficult to maintain and difficult to scale as more clients are signed up.

This company is currently in the process of re-architecting their solution around J2EE middleware and also wants to extract the process steps from the code to provide the type of flexibility, productivity, and maintenance benefits discussed earlier. The company has no requirement for persistence, other than logging to validate service response, since if a card is not validated within three seconds it is just swiped again. All interactions with the services in the process flow result in read-only interrogation of data, based on the input data, with routing rules applied to the data returned. The system itself must be able to handle, in the first instance, 30 of these end-to-end business transaction flows per second and must be able to scale effectively.

This scenario is difficult to fulfill for a traditional process vendor, as process tooling was never designed to deal with these types of flow scenarios, which are increasingly being referred to by a variety of terms – STP flows, microflows, and uninterruptible processes.

I will first show how you could go about achieving this with traditional process products, and then look at other tools that you can use to construct generated flows that do not reside in a process state engine.

"Previously it was possible to manually settle all trades, but the shortened cycle time, coupled with increasing volumes, meant that automation was the only real solution"

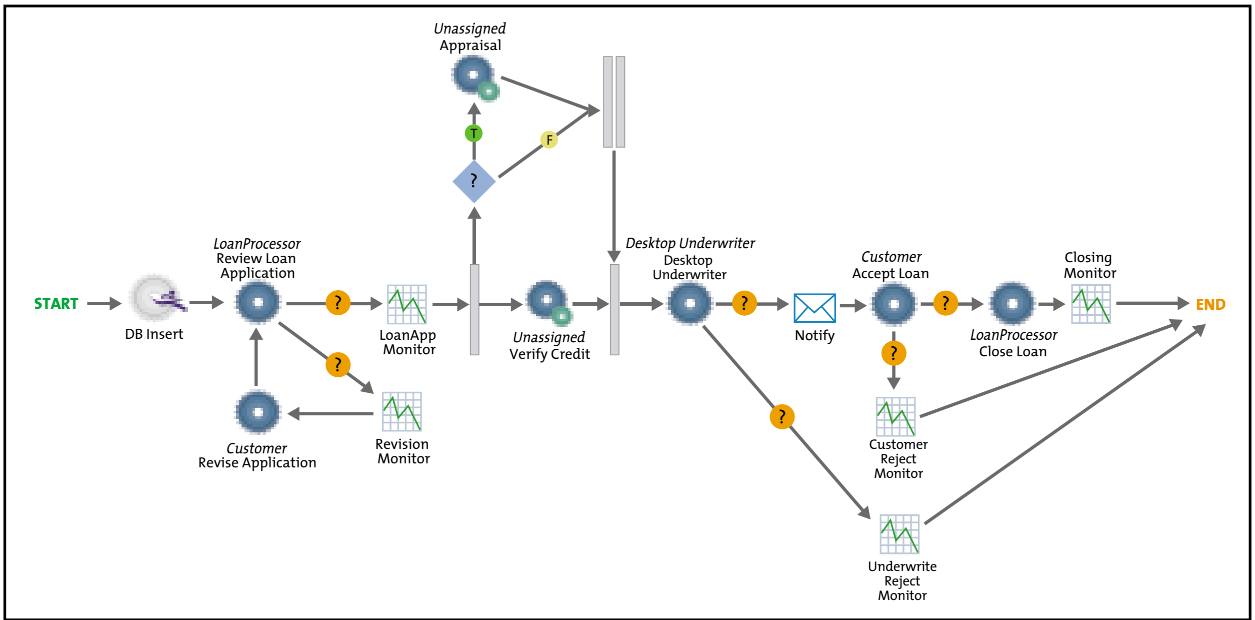


FIG. 1: LONG-LIVED ASYNCHRONOUS LOAN APPLICATION PROCESS

As discussed earlier, most process products persist their state to an underlying database. Even when using STP, the recording of initial and end state persistence means an I/O against the underlying database, which can add substantial overhead to an end-to-end flow in which the state engine database becomes the bottleneck. In our scenario we are not concerned with any recording of state, as in the event of a failure the process transaction is just restarted when the credit card is swiped again. Neither are we concerned with transaction management, as there are no transactions within the flow since all database activities are read-only.

Embedded databases provide a good solution when used with traditional process products to enhance STP performance and enable rapid execution of STP flows:

1. They can often be run in embedded mode as well as mixed mode.
2. There is no remote connection overhead in embedded mode.
3. They have an array of parameters to allow the tweaking of elements such as persistent file size and frequency of any disk I/O.
4. Because everything is executed in memory, they are extremely fast.

I recommend HSQLDB (http://hsqldb.sourceforge.net) as an open source embedded database. It is a good option for developers wishing to explore further the use of an embedded database for STP processes.

Construction of code-generated microflows is another way to approach STP flow requirements. WebSphere Studio Application Developer Integration Edition (v5) contains a plug-in that supports development with WebSphere's Process Choreographer. A business process container is configured for every WebSphere Application Server that has the Process Choreographer installed. The Process Choreographer contains support for what are referred to as interruptible and uninterruptible flows.

Interruptible flows are asynchronous. They execute in separate transactions and their state is stored in a database. Uninterruptible flows are visually described, resulting in generated code, with the meta-information of the process residing as FDML. These uninterruptible flows execute quickly either outside of a transaction or as a single transaction, and because each flow's steps execute in a single thread, they share the same context.

Although Process Choreographer uninterruptible processes do not require a state database, they do require the business process container that the Process Choreographer provides, as they make use of the choreographer engine.

Uninterruptible flow activities, or steps, can make use of snippets of Java code, as well as being able to invoke Web services and make calls to EJBs. The complete process can be exposed as a session bean facade or alternatively as a message-driven bean. This can then be packaged as a SOAP service if required. This is bound to an HTTP transport. However, using the Web Services Invocation Framework, you can also bind the session bean to an RMI or JMS transport at runtime.

This can all be done using wizards inside of WSAD IE, leading to a very productive and performant approach to creating straight-through process flows. The use of this tool and approach would be very applicable to the scenario I discussed earlier.

Figure 3 shows an example of an uninterruptible process that is provided with WSAD IE, and the associated Java snippet for the activity I will focus on.

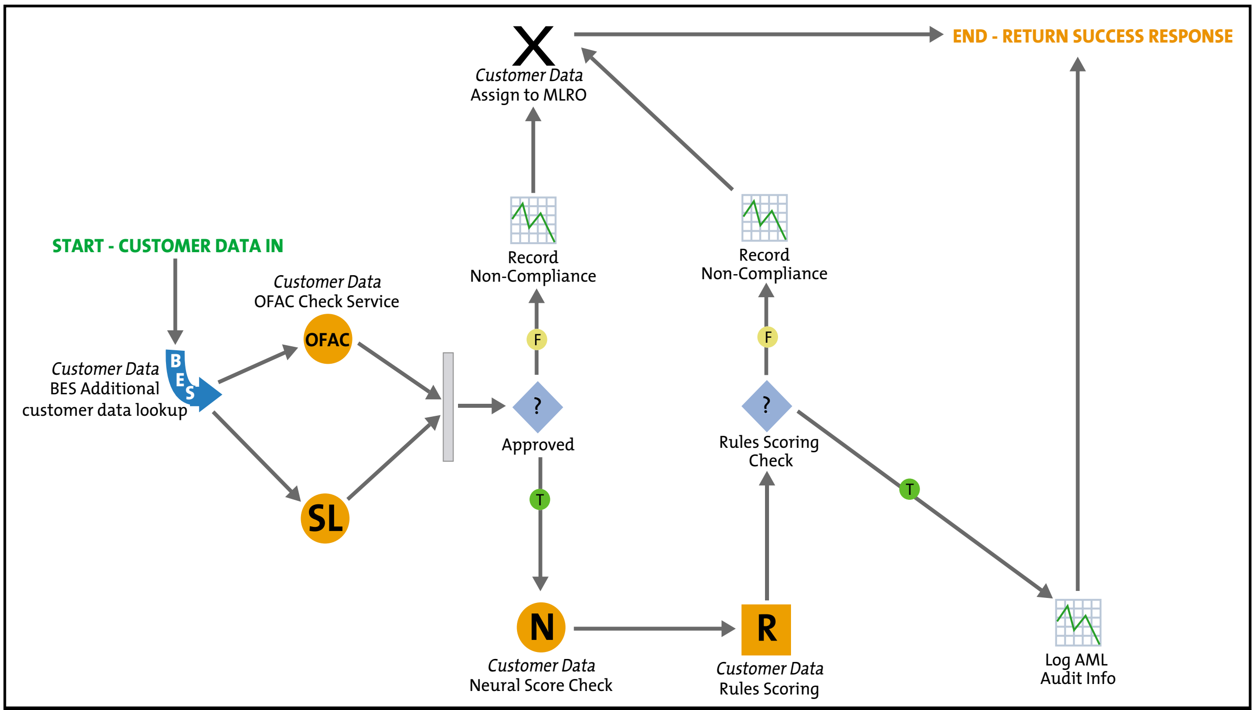
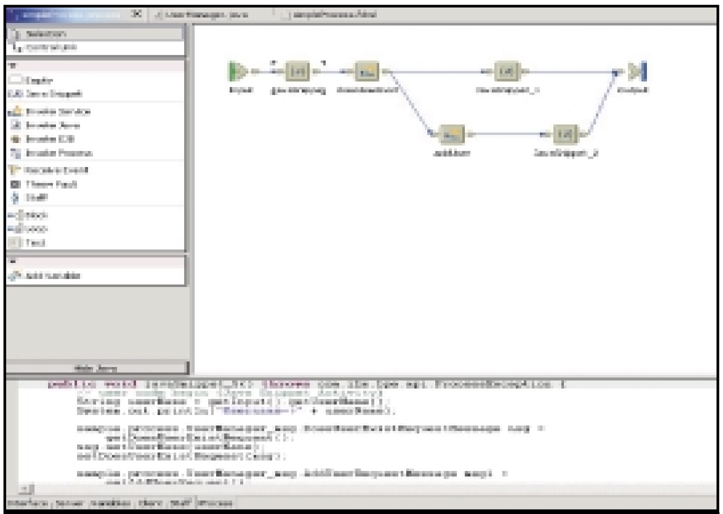

END - RETURN SUCCESS RESPONSE
Customer Data Assign to MLRO
START - CUSTOMER DATA IN
Customer Data BES Additional customer data lookup
Customer Data OFAC Check Service
OFAC
SL
Record Non-Compliance
Record Non-Compliance
F
F
Approved
Rules Scoring Check
T
T
N
Customer Data Neural Score Check
R
Customer Data Rules Scoring
Log AML Audit Info

FIG. 2: STP PROCESS AIMED AT PREVENTING MONEY-LAUNDERING


```
public void javaSnippet_3() throws com.ibm.bpe.api.ProcessException {
    // user code begin {Java Snippet Activity}
    String userName = getInput().getUserName();
    System.out.println("Executed-" + userName);

    sample.process.UserManager_msg.CheckUserExistRequestMessage msg =
        getCheckUserExistRequest();
    msg.setUserName(userName);
    setCheckUserExistRequest(msg);

    sample.process.UserManager_msg.AddUserRequestMessage msg1 =
        getAddUserRequest();
```

FIG. 3: AN EXAMPLE OF AN UNINTERRUPTIBLE PROCESS FLOW WITHIN WSAD IE

Process Definition for Java – JSR-207

Other initiatives exist that would support the construction of straight-through process flows, one being JSR-207, which is currently progressing through the Java Community Process. This aims to define metadata, interfaces, and a runtime model for creating business processes in the Java/J2EE environment. This proposal uses the Java Language Metadata technology (JSR-175) as a means to use a simple notation for describing business processes. The aim is for the metadata to be applied directly to Java source code in order to dynamically generate and bind process behaviors, which should include synchronous and asynchronous execution and support for straight-through flows. Anyone who has had cause to use XDoclet (http://xdoclet.sourceforge.net) – and to appreciate its simplicity and power – will appreciate this approach.

This is likely to be an important JSR, particularly because the functionality it provides is likely to find its way into the J2EE specification at some point. Further details can be found at www.jcp.org/en/jsr/detail?id=207.

Conclusion

To recap, a number of benefits can be derived from the use of STP within industries other than the financial industry with which it has traditionally been associated. This is especially pertinent to organizations that are embracing the concept of a loosely coupled and/or service-based architecture. These benefits include:

- Increase of business efficiency through the use of streamlined business processes
- Reducing cycle times and bottlenecks while improving operational efficiency
- Reducing manual intervention and therefore reducing costs
- Increasing productivity as the process steps are "extracted" from code, leading to easier design, maintenance, and change.
- Increased business visibility and information from process dashboards

Straight-though flows can be achieved today through the use of tooling and technology. This provides an effective way to increase business efficiency, visibility, and return on investment in the enterprise.

SORT IT OUT

WebSphere Technical Consulting

Making the most of your WebSphere investment is no easy task. MQSoftware's technical consultants can help. Our consultants are knowledgeable WebSphere Business Integration specialists with hands-on experience in diverse industries and technology environments. We'll bring you timely, cost-effective advice to minimize your design, installation and integration challenges, and we'll help you sort through the complexities of your middleware environment.

Call 800.998.2858 today or visit www.mqsoftware.com for more information.

MQSoftware™

IBM Business Partner

Expertise in the WebSphere Business Integration family of products.

Part 2: Putting theory into practice

Implementing J2EE/.NET Interoperability Using WebSphere MQ

BY BORIS **LUBLINSKY** AND DIDIER **LE TIEN**

In Part 1 of this series, we discussed how the use of messaging software can alleviate some of the problems with integration of J2EE and .NET environments using Web services. In this article we will discuss implementation of the proposed architecture on both J2EE and .NET platforms, along with possible enhancements of the proposed solution.

ABOUT THE AUTHOR
Boris Lublinsky is an enterprise architect at CNA Insurance, where he is involved in the design and implementation of CNA's integration strategy, building application frameworks and implementing SOA. Prior to joining CNA he was director of technology at Inventa Technologies, where he oversaw EAI and B2B integration implementations and development of large-scale Web applications. Boris has over 20 years of experience in software engineering and architecture.

E-MAIL
boris.lublinsky@cna.com

All of the code referenced in this article is available for download from www.sys-con.com/websphere/sourcec.cfm.

Implementing the .NET Client

Although support pack MA07 supports all the basic functionality of WebSphere MQ and is implemented based on the WebSphere MQ Java object model, implementation of interoperability with a JMS-based J2EE implementation poses the following challenges:

1. Implementation of JMS for WebSphere MQ introduces the RFH2 header, which is used by this implementation to support JMS-required features (e.g., custom properties). Messages sent using the JMS implementation contain the RFH2 header. The .NET implementation, on the other hand, treats the RFH2 header as part of the message payload.
2. Although there is standardized support for LDAP access in .NET, directory information for JMS is kept in the form of Java classes and cannot be interpreted by C# applications.
3. In the case of request/reply communications, matching of the replies to the appropriate requests is required. The most common technique for this is for the client to read the WebSphere MQ message ID (guaranteed to be unique) after the message is sent, and for the server to read this ID from the request message and put it into the correlation ID field of the reply message. In this case, a client can determine the corresponding request message based on the correlation ID of the reply message.

The two options for correlation ID matching use either the built-in WebSphere MQ or programmatic support. WebSphere MQ support allows reading from the specified queue based on the correlation ID. In this case only the message with the predefined correlation ID is read from the queue. This allows the client to send the message, get the message ID, and then do a blocking read with the timeout based on the correlation ID. The advantage of this approach is the programmatic simplicity. The drawback is that if the read times out, the message remains in the queue for a potentially long period of time. If the overall system performance slows down for some reason, the reply queue can start growing. This situation can be partially alleviated by limiting the "time-to-live" of the reply messages, thus implementing self-cleaning of the reply queue. This technique can be somewhat helpful, but it makes overall system monitoring more difficult because in this case messages disappear by themselves.

The first problem is solved by the creation of the JMSMessage class, derived from the MQMessage class, and encapsulating RFH2 header processing. Usage of the WebSphere MQ name resolution Web service solves the second problem. For the third problem, programmatic correlation is used.

We will begin by discussing the first two point solutions and then show how they are used in the overall client implementation.

IMPLEMENTING THE WEBSPHERE MQ OBJECT RESOLUTION WEB SERVICE

As mentioned above, the introduction of LDAP allows for a central point of control for the overall WebSphere MQ infrastructure. The problem is that the only existing standard for the LDAP entries supported by JMS defines all entries as Java classes, thus making it very hard to access LDAP information from other languages.

Our solution for this problem exposes LDAP information through an HTTP Web service implemented in

Java. This provides standardized access to the relevant LDAP information from any platform supporting HTTP Web services.

At first glance it may seem that using Web services for MQ object resolution defeats the purpose of our WebSphere MQ implementation. Instead of doing integration directly using Web services, we are now using Web services to resolve MQ objects, and then using MQ for the actual application's communications. In reality Web services are a supporting technology in this case, and their usage is fairly limited in our solution:

- The amount of data transferred by Web services is limited to the information on the MQ objects. The performance of this service can be well defined.
- Usage of caching, incorporated in our implementation, usually allows for minimizing overall Web services traffic.
- Because only one centralized Web service is used in the overall system, this service can be optimized for high availability.
- All of the quality-of-service aspects of the actual consumer-to-provider communications are defined by WebSphere MQ–based communications rather than by the Web services.

As shown in Figure 1, the overall implementation is composed of two parts: implementation of the service itself, and implementation of the .NET client. This includes the Web service client itself, a caching mechanism, and an interface to the rest of the implementation.

SERVICE IMPLEMENTATION

As mentioned earlier, the actual service is implemented in Java in the form of a JavaBean that exposes two methods:

- ***MQData serviceLookUp (String servicename):*** This method retrieves the information about the producer WebSphere MQ environment, including request queue information as well as information about the producer queue manager. The reply is sent back within a single MQData structure in order to minimize the network traffic and the number of method invocations.
- ***MQData consumerLookUp (String consumername):*** This method retrieves the information about the consumer WebSphere MQ environment, including reply queue information as well as information about the consumer queue manager. The reply is sent back within a single MQData structure in order to minimize the network traffic and the number of method invocations.

Here MQData is the generic data structure class (see Listing 1) representing both queue and queue manager information. It is populated by the JNDILookUpUtil class (see Listing 2). The consumername or servicename string defines the consumer/producer name in a standard JNDI format, i.e., /Path1/Path2/Path3. The conversion to the complete JNDI location, along with the conversion to the format of the specific JNDI implementation, is done under the covers by a dedicated class called ServiceQueueString (see Listing 3). Usage of the standard JNDI format allows our implementation to support any JNDI implementation (LDAP-based, file-based, or WAS internal JNDI implementations are currently supported). Bean implementation (see Listing 4) is configurable through the configuration file (see Listing 5) containing the provider URL and the Initial Context Factory.

Standard WSAD 5 Web services support was used to create the Web service from our JavaBean. The WSDL files generated by WSAD are presented in Listings 6–9.

SERVICE CLIENT IMPLEMENTATION

The service client is implemented as .NET class library. Our implementation uses the Web service proxies and MQData class automatically generated by Visual Studio .NET from the Service.wsdl and Binding.wsdl files produced by WSAD 5.0.

Implementation includes the JNDIProxy and JNDICache classes. The JNDIProxy (see Listing 10) is used by any application that accesses the JNDI Web service. The following methods are exposed:

- ***public MQData ProducerLookUp (string producername):*** This method returns an MQData structure that includes the information related to the producer WebSphere MQ environment. Only input queue information is used by the .NET client implementation.
- ***public MQData ConsumerLookUp (string consumername):*** This method returns an MQData structure that includes information related to the consumer WebSphere MQ environment, including consumer queue manager information and consumer reply queue name.

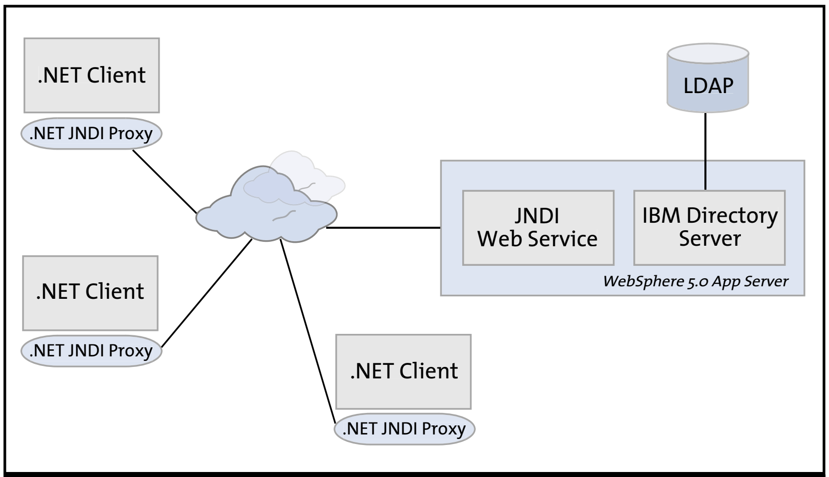
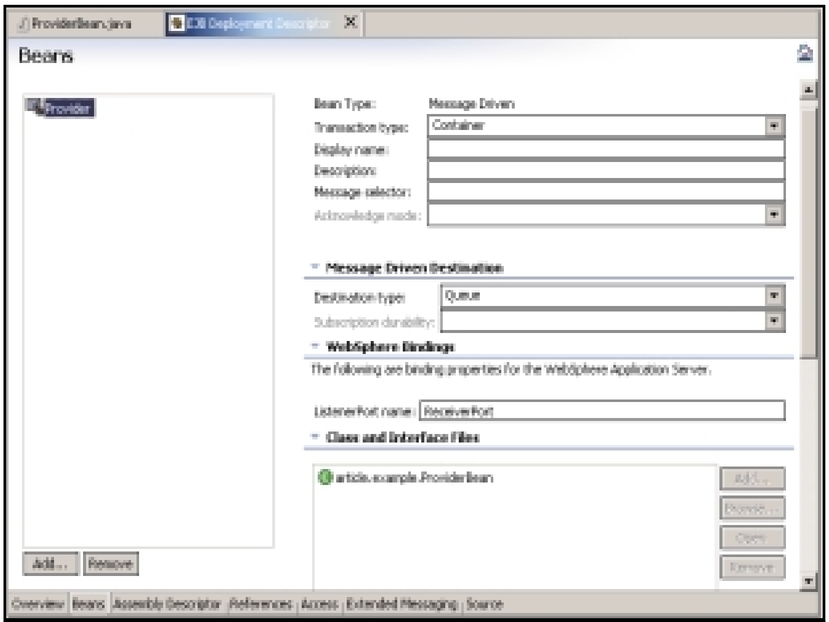


FIG. 1: JMS MQ LOOKUP SERVICE ARCHITECTURE

ABOUT THE AUTHOR

Didier Le Tien is an enterprise architect at CNA Insurance, where he is involved in the design and implementation of CNA's integration strategy, building application frameworks and implementing service-oriented architecture for the company. He holds a PhD in computer science from the University of Evry, in France.

E-MAIL
didier.letien@cna.com


ProviderBean.java | EJB Deployment Descriptor

Beans

Provider

Bean Type: Message Driven
Transaction type: Container
Display name:
Description:
Message selector:
Acknowledge mode:

Message Driven Destination
Destination type: Queue
Subscription durability:

WebSphere Bindings
The following are binding properties for the WebSphere Application Server.
ListenerPort name: ReceiverPort

Class and Interface Files
article.example.ProviderBean

FIG. 2: DEFINING A LISTENING PORT FOR THE PROVIDER

In order to limit the amount of communication between the JNDIProxy class and the JNDI Web service, each invocation result is stored (based on the consumer/producer name) in the JNDICache class (see Listing 11). Before invoking the JNDI Web service, the JNDIProxy checks its cache to see whether or not an entry has already been processed. If the entry exists and is not too old (a reference is considered "old" when its lifetime exceeds three hours), then the proxy returns that value to the invoker. Otherwise, a Web service is invoked.

IMPLEMENTING THE JMSMESSAGE CLASS

IBM introduced the RFH2 header to support JMS-specific data. The RFH2 header is extensible and can also carry additional information that is not directly associated with JMS. For the purpose of our discussion, only JMS-specific usage of the RFH2 header is covered.

The header is composed of two parts, one fixed, and the other variable. The fixed portion of the header contains such information as message format, encoding, flags, etc. The JMS-specific variable portion of the header can contain the following three folders:

- ***The <mcd> folder:*** Contains properties that describe the shape or format of the message. For example, the Msd property identifies the message as being Text, Bytes, Stream, Map, Object, or "Null". This folder is always present in a JMS RFH2 header.
- ***The <jms> folder:*** Used to transport JMS header fields and JMSX properties that cannot be fully expressed in the MQMD. This folder is always present in a JMS RFH2 header.
- ***The <usr> folder:*** Used to transport any application-defined properties associated with the message. This folder is only present if the application has set some application-defined properties.

Basic RFH2 processing is implemented by the RFH2 class, shown in Listing 12.

The MA7P support pack being used for WebSphere MQ programming appends RFH2 content to the message payload. This means that the RFH2 content has to be extracted from the incoming message and appended to the outgoing one. This functionality is implemented by the JMSMessage class (see Listing 13). The interface for this class was modeled after the Message interface in JMS.

Overall Client Implementation

The overall WebSphere MQ communication client is implemented as a class library, exposing the RequestProcessor class (see Listing 14). This class exposes three methods to its users:

- ***RequestProcessor getRequestProcessor(string producer):*** This static method serves as a RequestProcessor factory, allowing for the creation of a new instance of the class, configured to communicate to the specific producer. Besides creating and returning the reference of the new RequestProcessor instance, this method ensures proper initialization of the MQ environment and other required classes. Configuration parameters for a particular instance are retrieved from the Config.config file (see the sample config file shown in Listing 15), which has to be located in the same directory as the client application itself.

  The ConfigurationParameters class (see Listing 16) is used to read and process configuration parameters. The set of configuration parameters includes:
  –*wsURL:* The URL for the Web ser–vice, which supports JNDI lookups.
  –*name:* The name of the client. This value is used to request MQ environment information from the JNDI lookup Web service.
  –*receiverThreads:* The number of threads that the MessageReceiver class will run. The getRequestProcessor method also ensures that the message receiver is correctly initialized. It uses the JNDI lookup Web service to obtain information about the queue that the producer is listening on.

- ***string requestReply(string message, int timeout):*** This method executes request/reply (blocking) for the producer defined in the factory method. It takes two parameters – request message (string) and timeout (in milliseconds). It returns a reply message (string) or, if communication fails, throws an exception.

MQSoftware
SEE IT WORK.™

8,557 orders

flowed through your system flawlessly today.

1 didn't.

Unfortunately, one failed transaction a day

will cost you $4.3 million this year.

– If only you saw it sooner.

.::::..:.:: Real-time monitoring of all your critical business transactions.

:.'':.::.:.::'. Because finding out tomorrow might be too late.

MQSoftware
Q Nami!™

IBM Business Partner

To learn more about Q Nami! and the benefits it brings to your business, register to attend a free online Web seminar.
Visit **www.mqsoftware.com/qnami** for seminar dates, or call **800.998.2858** for details.

- ***void fireAndForget(string message):*** This method sends a request to the producer defined in the factory method and immediately returns. It takes a single parameter – request message (string). The method throws an exception if the sending of a message fails.

Two classes implement the actual WebSphere MQ communications:

- ***MessageSender:*** This class (see Listing 17) is a straightforward implementation of the MQ message sender with the following caveats:
  –Because in .NET implementation of the MQQueueManager object is not transferable between threads, it is necessary to create a new instance of this object (MQ connection) on each thread (for each instance of the class).
  –In the case of fire-and-forget, persistent messaging (the default) is used. In the case of request/reply we use nonpersistent messages (which results in significant performance improvement). Also, in this case we are setting an expiration time for messages (making the queue self-cleaning).
  –We are using our own coordination (described below) to coordinate between request and reply messages.

- ***MessageReceiver:*** This class (see Listing 18) is a multithreaded implementation of the message receiver.

Coordination between MessageSender and MessageReceiver is implemented using two classes:

- ***DataRetriever:*** This class (see Listing 19) supports dual functionality:
  –Data transfer between sender and receiver threads. This is done through the private reply object, which is set by the receiver thread and read by the sender thread.
  –Synchronization between sender and receiver threads, based on the Monitor object (standard synchronization support in .NET).

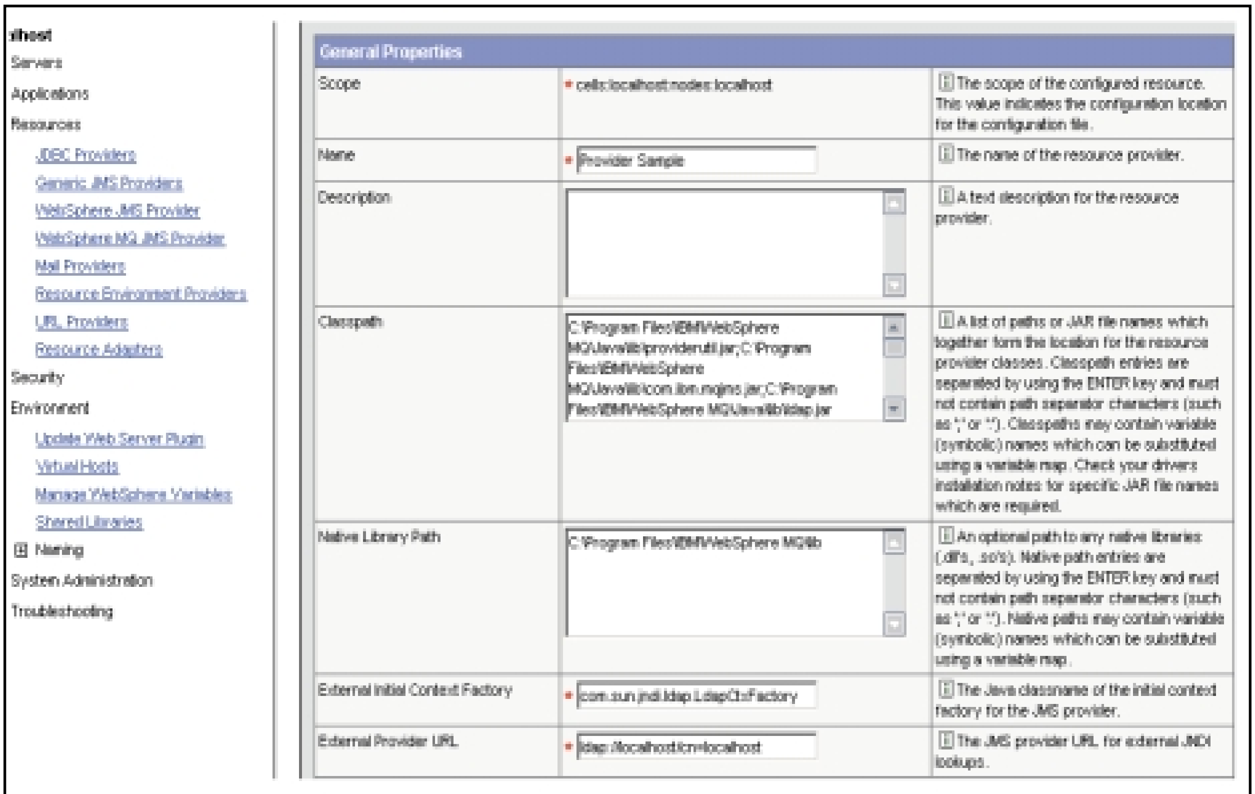
- ***Correlator:*** This class (see Listing 20) is a thread-safe wrapper around a standard hashtable class (not thread-safe in .NET). A unique message ID/correlation ID, generated by MQ and set by the provider implementation, is used as a key between sender and receiver threads.

Implementing the J2EE-Based Server

There are many approaches to implementing a J2EE-based consumer using messaging. We have chosen a message-driven bean (MDB)-based implementation of the producer.

CREATING THE PRODUCER IMPLEMENTATION

A standard pattern for implementing applications using MDBs is to make the MDB implementation responsible for messaging and invoking a stateless entity bean for the actual message processing (business functionality). Due to the extreme simplicity of our implementation, we are not following this practice, but rather implementing a separate method on the same bean for the actual processing. The functionality of this method is fairly straightforward: it takes a string from the incoming



General Properties		
Scope	* cells:localhost:nodes:localhost	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* Provider Sample	The name of the resource provider.
Description		A text description for the resource provider.
Classpath	C:\Program Files\IBM\WebSphere MQ\Java\lib\providerutil.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\ldap.jar	A list of paths or JAR file names which together form the location for the resource provider classes. Classpath entries are seperated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Classpaths may contain variable (symbolic) names which can be substituted using a variable map. Check your drivers installation notes for specific JAR file names which are required.
Native Library Path	C:\Program Files\IBM\WebSphere MQ\lib	An optional path to any native libraries (.dll's, .so's). Native path entries are seperated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths may contain variable (symbolic) names which can be substituted using a variable map.
External Initial Context Factory	* com.sun.jndi.ldap.LdapCtxFactory	The Java classname of the initial context factory for the JMS provider.
External Provider URL	* ldap://localhost/cn=localhost	The JMS provider URL for external JNDI lookups.

FIG. 3: DEFINING A GENERIC JMS PROVIDER

Introducing JProbe® 5.0

...now smarter and faster than ever

JProbe®

Find the cause of J2EE code performance, memory and threading problems faster than ever before with JProbe 5.0. New investigative features for finding memory problems combined with dramatic performance improvements mean even quicker resolution of problems in your application, servlet, JSP and EJB code.

JProbe Suite

JProbe Profiler

JProbe Memory Debugger

JProbe Threadalyzer

JProbe Coverage

software development 13th annual productivity award

QUEST SOFTWARE™

For more info and a free eval, visit:

http://java.quest.com/jprobe/wdj

message and appends it to the “Hello” string. It then sends a response back using the ReplyToQ information specified by the consumer.

WSAD v5 tooling was used for the generation of the MDB itself, and the required additional functionality was added manually. The MDB code is shown in Listing 21. After its creation, the MDB must be associated with the listening port (see Figure 2).

DEPLOYING THE PRODUCER APPLICATION ON WAS

In order to deploy the MDB, the following must be done:

- WAS messaging must be configured to define the queue and queue manager that the MDB is listening on.
- The listener port must be configured, tying this bean to the queue/queue manager definition created in the previous step.

Configuration of the queue/queue managers’ information is done through the administrative console client. WAS v5 allows three different JMS “provider” configurations for messaging:

1. ***WebSphere JMS provider:*** A simple, internal JMS server provided with WAS for messaging to support the J2EE 1.3 requirement for messaging. All configuration for this can occur within the WAS admin console (or WSAD). The JMS JNDI entries are defined in the WebSphere JNDI namespace.
2. ***WebSphere MQ JMS provider:*** Alternatively, you can continue to use WebSphere MQ for messaging. The administration of this is greatly improved in WAS 5, because all configuration for this provider can occur within the WAS admin console (or WSAD). The JMS JNDI entries in this case have to be defined in the WebSphere JNDI namespace.
3. ***Generic JMS provider:*** An alternative that allows the ability to configure external providers for JMS messaging, including the use of external JNDI registries for the JMS entries. External messaging server providers are also provided.

The use of an external LDAP directory in our architecture implies usage of a generic JMS provider. Use of a generic provider requires definition of the following:

- Provider name
- JAR files used by our JMS provider (in our case, the WebSphere MQ JAR files)
- Native library path used by our provider (WebSphere MQ path)
- External context factory (in our case LDAP) and the LDAP provider URL

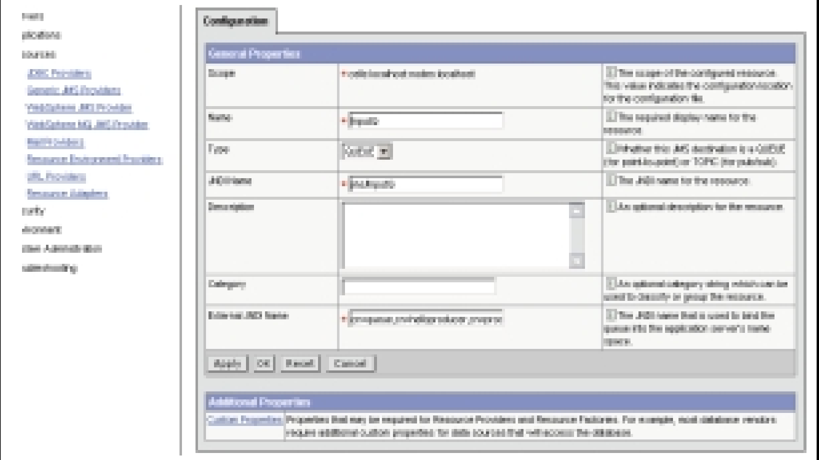
The configuration screen for a generic messaging provider is shown in Figure 3.

In addition, resources (i.e., queues and queue connection factories) must be defined in order for the provider to work.

The producer queue is defined by creating a “JMS Destination,” which can be set as one of our generic JMS provider properties. The process consists of defining a local JNDI reference, mapping it to an LDAP entry, and specifying the type of the entry (i.e., Queue or Topic). In our case we are defining a queue type. A sample JMS destination screen is shown in Figure 4.

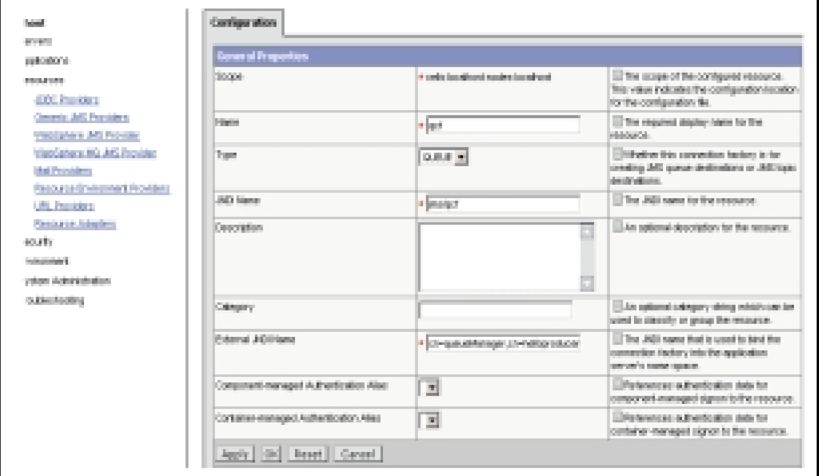
Similarly, a queue connection factory must be defined for a generic JMS provider, including local JNDI reference, external reference, and type (queue, in our case). A sample JMS queue connection factory screen is shown in Figure 5.

Once the JMS destinations and the JMS connection factory are defined, we can then define the input port used by our MDB implementation to pull the messages out of the queues.



Scope: cells:localhost:nodes:localhost
Name: InputQ
Type: QUEUE
JNDI Name: jms/InputQ
Description:
Category:
External JNDI Name: [illegible]

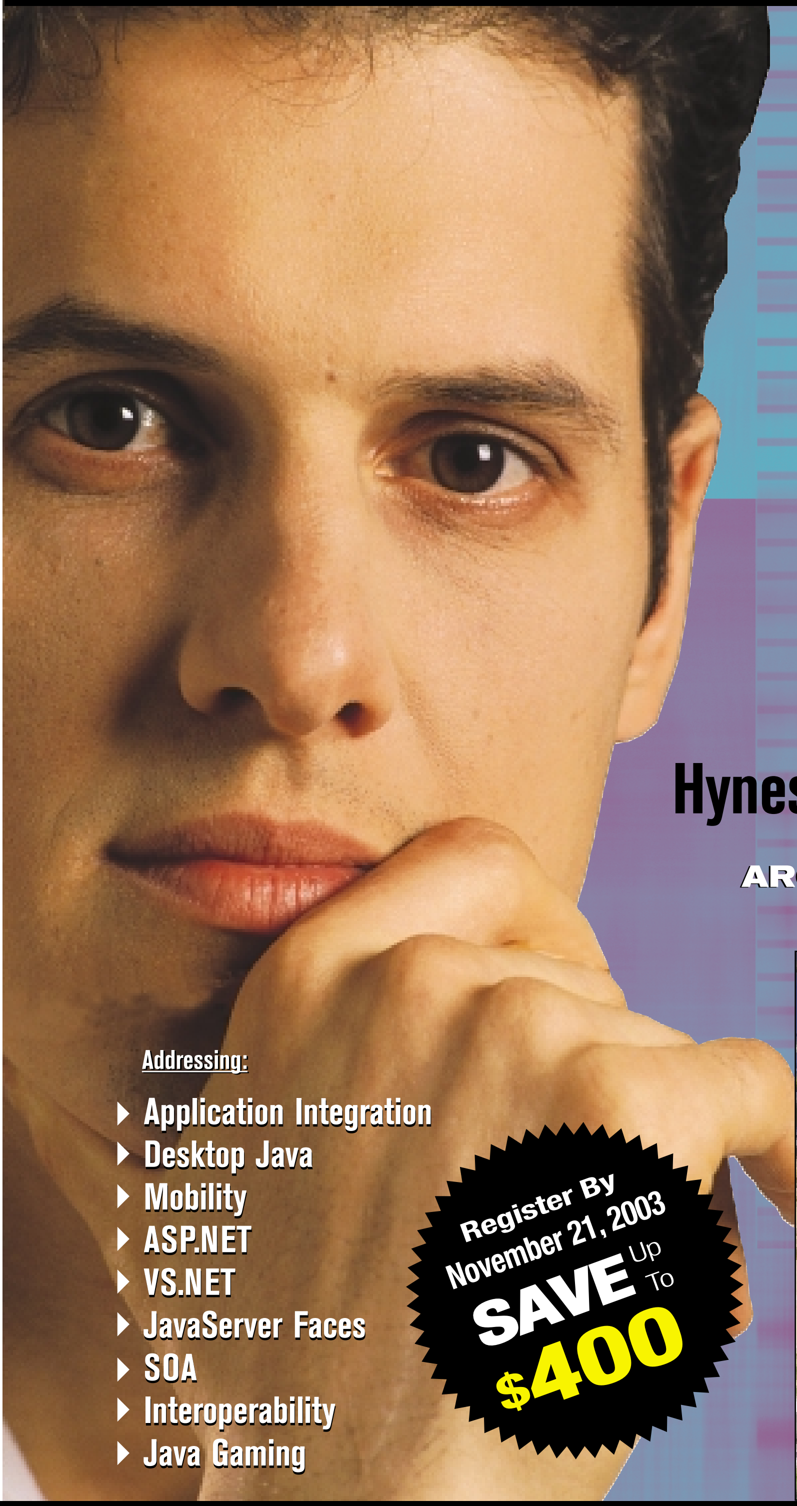
FIG. 4: DEFINING THE INPUT QUEUE



Scope: cells:localhost:nodes:localhost
Name: qcf
Type: QUEUE
JNDI Name: jms/qcf
Description:
Category:
External JNDI Name: cn=queueManager,[illegible]
Component-managed Authentication Alias:
Container-managed Authentication Alias:

FIG. 5: DEFINING A QUEUE CONNECTION FACTORY

International Conference & Expo



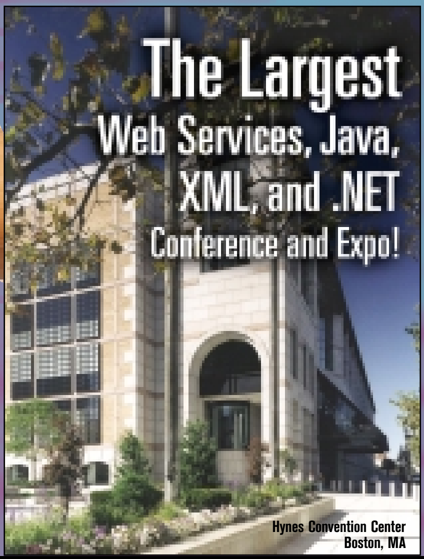
Edge 2004 EAST

Development Technologies Exchange

February 24-26, 2004

Hynes Convention Center, Boston, MA

ARCHITECTING JAVA, .NET, WEB SERVICES, OPEN SOURCE, AND XML


The Largest
Web Services, Java, XML, and .NET Conference and Expo!
Hynes Convention Center Boston, MA

Addressing:

- Application Integration
- Desktop Java
- Mobility
- ASP.NET
- VS.NET
- JavaServer Faces
- SOA
- Interoperability
- Java Gaming

Register By November 21, 2003 SAVE Up To $400

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

Over 200 participating companies will display and demonstrate over 500 developer products and solutions.

Over 3,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 100 of the latest sessions on training, certifications, seminars, case-studies, and panel discussions promise to deliver real world benefits, the industry pulse and proven strategies.

Full Day Tutorials Targeting

- Java • .NET • XML
- Web Services • Open Source

Conference program available online!
www.sys-con.com/edgeeast2004

Contact information: 201 802-3069 • events@sys-con.com

OWNED BY SYS-CON MEDIA · LinuxWorld Magazine · LinuxWorld.com · Java Developer's Journal · JavaWorld · WebServices Journal · Linux Business Week · XML Journal · OASIS · SD Times · eaIndustry Consortium · PerfectXML

PRODUCED BY SYS-CON EVENTS · .NET Developer's Journal · asp.netPRO · WebSphere Developer's Journal · WebLogic Developer's Journal · wireless Business & Technology · CF Advisor · ColdFusion Developer's Journal · ASPstreet.com · HSP Street.com · PowerBuilder Developer's Journal · MX developer's journal

Testing Our Implementation

In order to verify that our implementation works correctly, we created a very simple C# client application (see Listing 22) using our .NET client implementation. This client contains a simple form (see Figure 6) that contains two text boxes (one for entering a request and another for displaying the response), two labels, and a button for submitting the request to the MQ Client. Our implementation was generated automatically by Visual Studio .NET, except for the Sbutton_Click(object sender, System.EventArgs e) method implementation. This method invokes our MQ client to synchronously invoke the J2EE-based provider. The result of the test client execution is shown in Figure 7.

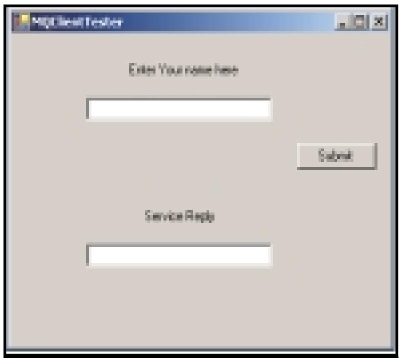
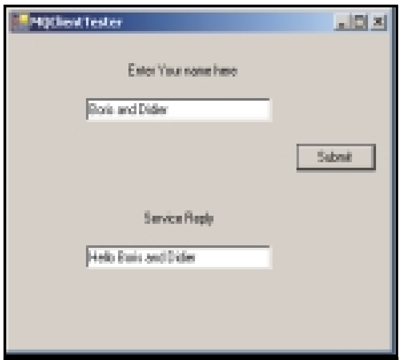


FIG. 6: TEST CLIENT FORM



Boris and Didier

Hello Boris and Didier

FIG. 7: TEST CLIENT EXECUTION RESULT

Additional Options for Using Web Services Over MQ

The example presented here describes only basic WebSphere MQ–based messaging implementation between .NET-based and J2EE-based application servers. Real-life implementations should consider messaging formats between applications. One of the most ubiquitous data communications standards today is SOAP, which is supported extremely well by both .NET and J2EE application servers. A completely revised version of IBM's support pack – MA0R (www-3.ibm.com/software/ integration/support/ supportpacs/ individual/ ma0r.html) – addresses just that. This support pack provides an implementation for both the .NET and J2EE sides to implement SOAP over MQ using Web services interfaces.

The .NET implementation is based on the overwriting of the .NET class ApplicationHost, enabling the host to create application domains for processing ASP.NET requests by a specialized implementation supporting both HTTP and WebSphere MQSeries requests. In the case when .NET acts as a service producer, this implementation provides both MQ and HTTP listeners that allow for the receiving of SOAP messages using either transport, and invoking a .NET SOAP-processing pipeline for processing of incoming SOAP messages. If .NET is used as a service consumer, the provided implementation uses either MQ or HTTP transport for request/reply messages based on the URL of the service.

The J2EE implementation is based on Axis (http://ws.apache.org/axis) from the Apache foundation and supports WebSphere MQ (JMS). Combining an AXIS server with the MDB implementation described here allows a J2EE server to effectively receive and process SOAP messages over MQ. Based on the Web service URL, an Axis-based client can use either HTTP or JMS as a client.

Usage of SOAP over MQ for .NET/J2EE interoperability allows combining standards-based Web services with the MQ messaging infrastructure widely used in many companies today.

Conclusion

Selecting WebSphere MQ as an interoperability solution between .NET and J2EE-based application servers allows the leveraging of a proven technology that already exists in many enterprises. In comparison with the Web services approach, the MQ-based solution provides the following advantages:

- Reliable messaging
- "Out of the box" support for asynchronous messaging (not described in this article) that allow for interactions with temporarily unavailable applications
- Simpler scalability implementation through producer implementation "cloning" and MQ clustering (not described in this article)
- Better application decoupling

Usage of MA07, although not currently supported by IBM, provides support for the merging of Web services technology with reliable messaging that leverages a widely adopted, standard, rich communication paradigm over a well-established, reliable transport.

Acknowledgments

We are thankful to our colleagues at CNA and IBM, especially Matt Segvich, Vallish Shankar, Gerald Daus, and Deborah Shaddon for their help in putting together this article and supporting code.

"One of the most ubiquitous data communications standards today is SOAP, which is supported extremely well by both .NET and J2EE application servers"

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO $400 AND RECEIVE UP TO 3 FREE CDs!*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

3-Pack
Pick any 3 of our magazines and save up to $275⁰⁰
Pay only $175 for a 1 year subscription plus a FREE CD
- 2 Year – $299.00
- Canada/Mexico – $245.00
- International – $315.00

6-Pack
Pick any 6 of our magazines and save up to $350⁰⁰
Pay only $395 for a 1 year subscription plus 2 FREE CDs
- 2 Year – $669.00
- Canada/Mexico – $555.00
- International – $710.00

9-Pack
Pick 9 of our magazines and save up to $400⁰⁰
Pay only $495 for a 1 year subscription plus 3 FREE CDs
- 2 Year – $839.00
- Canada/Mexico – $695.00
- International – $890.00

CALL TODAY! 888-303-5282

Pick a 3-Pack, a 6-Pack or a 9-Pack

☐ 3-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 6-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 9-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.

TO ORDER •Choose the Multi-Pack you want to order by checking next to it below. •Check the number of years you want to order. •Indicate your location by checking either U.S., Canada/Mexico or International. •Then choose which magazines you want to include with your Multi-Pack order.

☐ **MX Developer's Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $167 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $29.99 /	Save: $60
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $137 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $40
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $127 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $30
Digital Edition - One Year (12)	You Pay: $19.99	

☐ **Linux World Magazine**

U.S. - Two Years (24) Cover: $143	You Pay: $79.99 /	Save: $63 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Java Developer's Journal**

U.S. - Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Web Services Journal**

U.S.- Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **.NET Developer's Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **XML-Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **WebLogic Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

☐ **ColdFusion Developer's Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int'l - Two Years (24) $264	You Pay: $189 /	Save: $75 + FREE $198 CD
Int'l - One Year (12) $132	You Pay: $129.99 /	Save: $2

☐ **Wireless Business & Technology**

U.S. - Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $192	You Pay: $139 /	Save: $53 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $79.99 /	Save: $16
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **WebSphere Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

☐ **PowerBuilder Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Subscribe Online Today www.sys-con.com/2001/sub.cfm

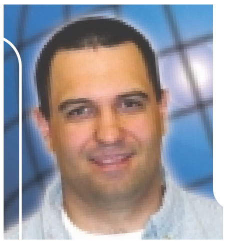
SYS-CON MEDIA

Software configuration management makes it easy

Teamwork in WebSphere Studio

BY ADAM **TERRENZIO**

So what does teamwork in WebSphere Studio mean anyway? At one level it means being able to work on code within a shared environment through which your application can be executed and tested. WebSphere Studio already has great support for the deployment and testing of such shared development. At another level, however, it means being able to share the development effort itself.



ABOUT THE AUTHOR

Adam Terrenzio is a lead application developer on the product integration team at MKS Inc., an enterprise software configuration management vendor (www.mks.com), primarily responsible for the IBM WebSphere Studio products and other IDE integrations. Adam works at the MKS worldwide headquarters in Waterloo, Ontario, Canada.

E-MAIL

adam.terrenzio@mks.com

Software configuration management (SCM) is one means through which development effort can be shared. Even prior to deployment and testing, developers can become aware of other developers' changes and assess their impact. With a good SCM tool, this form of interaction can take place between developers even in separate geographic locations and time zones – all in a controlled manner. Along with actual code changes, the history and the design reasoning can be preserved, and there is an overall improved sense of communication between all developers.

State of Play

As the adoption rate of WebSphere Studio tools continues to climb, so do the number of plug-ins available from independent software vendors (ISVs). This highly flexible and adaptable interface has allowed everyone who ever said "I wish it could do this, in this fashion" to go ahead and add useful and productive tools to the WebSphere Studio family of products.

See the Resources section for the link to IBM's Plug-in Central, which offers a complete list of IBM "Ready For WebSphere" Certified plug-ins.

Some of the earliest adopters of WebSphere Studio were SCM companies. Historically, integrated development environments have had many different ways to access source code controlled in repositories. Some of these methods include standard and custom APIs or script/command line–based access. All SCM integrations out there do pretty much the same thing in very similar ways. They allow complex development teams to manage and develop multiple projects by sharing the code base in a controlled manner. Developers have access to the tools they need to perform their day-to-day activities within the WebSphere Studio interface, reducing the amount of application and window switching. This helps them stay organized and efficient because they can get all of their information on the projects they are working on from within one tool. So, how do SCM plug-ins help developers?

Access to SCM Commands

While exploring the WebSphere Studio context menus you may have noticed a *Team* menu. You probably just scratched your head and thought, "I wonder what this is for? All of the options are grayed out anyway." "Team" is what the WebSphere Studio interface calls source configuration management. Once you have installed an SCM plug-in, this is where all of your source code control menu items will be located. Most likely the list includes Check Out, Check In, Add, Drop, Resynchronize, and many others (see Figure 1).

Many providers will also include a menu in the main toolbar, or a set of buttons to access the tool. As with other features of WebSphere Studio, this is totally configurable on a per-perspective basis. For more information, see the WebSphere Studio help on how to customize a perspective.

Text/Label Decorators

Who doesn't love pretty little pictures? If a plug-in offers this feature (see Figure 2), it means that it enhances (or replaces) the existing file icons in any view that shows resources (files, folders, and projects). There is also the capability to append the filename with more information in a textual format. The types of questions this information can answer include:

- Do I have the most recent version of this file?
- If so, what version am I working on?
- If not, how far behind am I?
- Is the file locked?
- If so, by whom?
- Does my local file differ from the archived version?

This information is helpful for many reasons. Perhaps you run a build of your project and get some compile errors in files you haven't touched recently. Perhaps those files are simply out of date because another user has changed them. You can quickly see which files have changed in your workspace and evaluate which you may (or may not) need to resynchronize to get a successful build. Locking a file (which will show in other users' workspaces) informs developers that they may wish to wait to make changes to a specific file.

There are also different locking styles that SCM vendors use. One of these models is optimistic locking. You can get any file at any time and conflicts are resolved on check-in by performing merges. Optimistic models require what is known as "conflict resolution."

The contrast to this is pessimistic locking, which means that once a file is locked by a user, all other users are made aware ahead of time that if they make a change to that file they will need to do a merge later. A pessimistic model is often referred to as a "conflict avoidance" approach. Some SCM tools allow you to see what you have locked, as well as what files are currently locked by other users. This allows the user to decide whether to perform a merge once changes have been made, or to just wait until the file is checked in by the previous user and then make changes (see Figure 2).

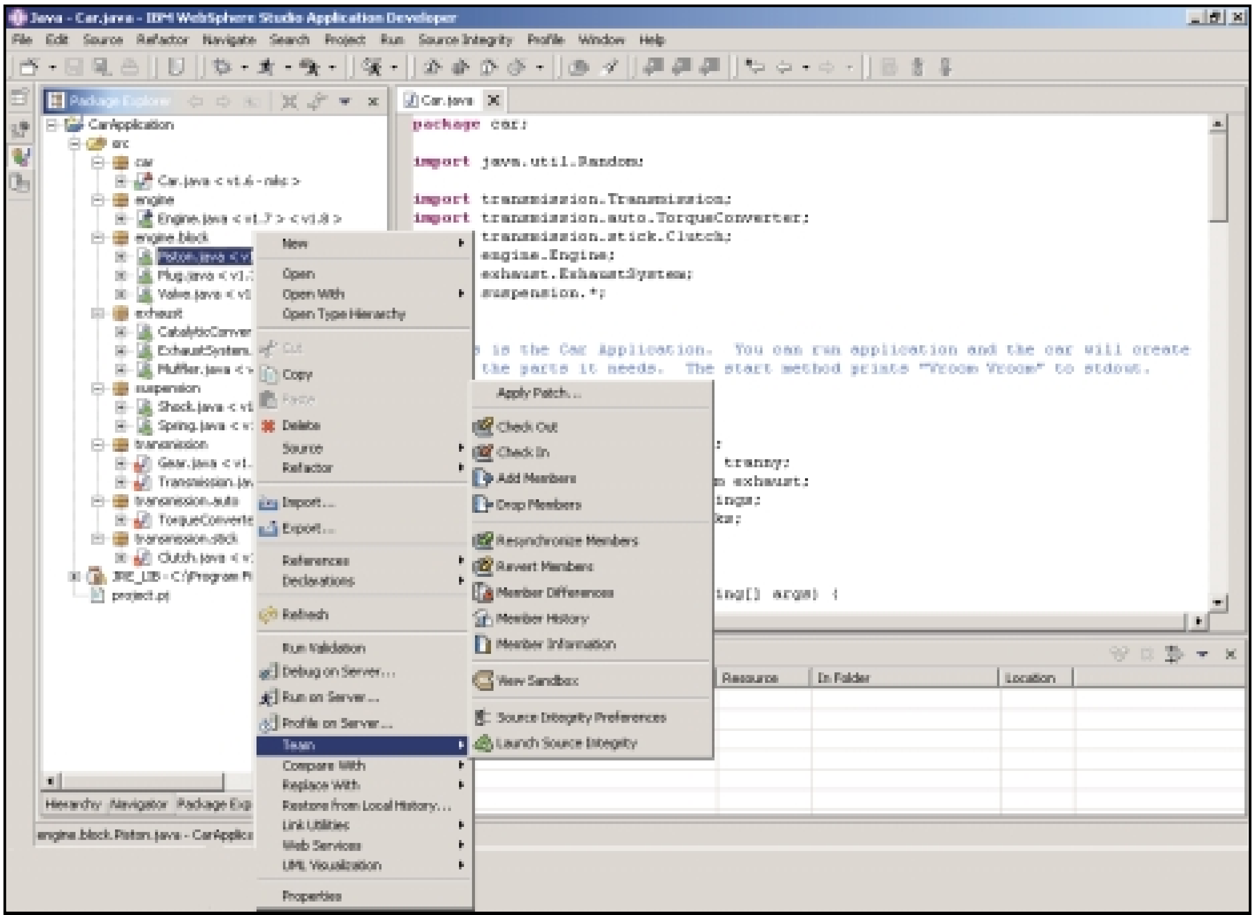
Team Project Sets

The *Import* item under the *File* menu houses many different ways to get files, folders, and projects into the workspace. One of the options is to import a *Team Project Set*. What this command allows you to do is to import an entire workspace at one time. So if you have a complicated application structure with many related projects, you can import all of them at once! This is an extremely powerful feature that can be used to replicate workspaces on any computer in a single step.

The way this works is your team lead sets up his or her workspace and all of the project relationships, classpath, compile options, and other WebSphere Studio–specific project information. Then they create a Team Project Set file (.psf extension) from the File menu by choosing Export. This file is really just XML-formatted instructions that tell the application how to re-create the workspace. Don't worry if you don't see everything in there that you think it should have. This information is cryptic and meant to be read by the application and not by the user. All of the project metadata and relationship information is kept in the .project file that WebSphere Studio creates for each project. When the workspace is populated during an import, all of that information is still there because the .project file is archived.

Preferences Are Your Friends

There is also a section labeled *Team* in the main WebSphere Studio preferences panel. Here you can find some generic preferences that can affect how SCM plug-ins behave, as well as specific team provider preferences. Many SCM vendors allow you to tell the backing



```
package car;

import java.util.Random;

import transmission.Transmission;
import transmission.auto.TorqueConverter;
       transmission.stick.Clutch;
       engine.Engine;
       exhaust.ExhaustSystem;
       suspension.*;


 is the Car Application.  You can run application and the car will create
 the parts it needs.  The start method prints "Vroom Vroom" to stdout.
```

FIG. 1: SCM COMMAND MENU FROM MKS SOURCE INTEGRITY ENTERPRISE EDITION INTEGRATION WITH WSAD

repository how to store files. The two most common options are as text files and as binary files. The *File Content* preference page allows you to specify which of the two options you would like to use based on file extension.

Another vendor-generic page is the *Ignored Resources* page. This page allows you to use regular expressions to specify files or directories that you would like the SCM system to ignore completely. Common things to specify here are bin directories and test file extensions (.bak, for example). This can be an incredibly useful tool when performing large operations.

Some of the vendor-specific information that may also be here includes things such as connection and server information, and behavior specifics. You may choose to have every action in the workspace echoed directly in the repository, for example.

Conclusion

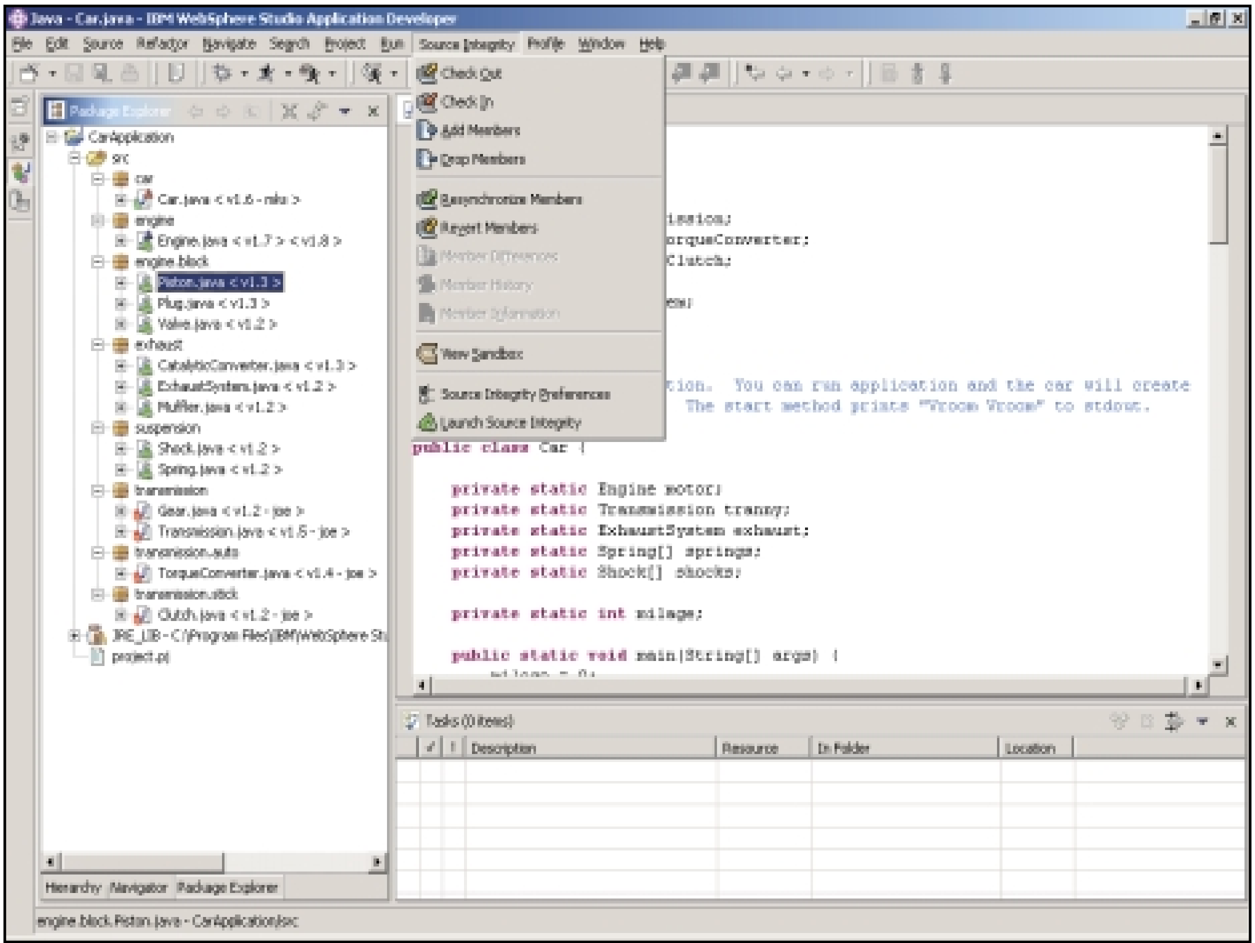
SCM plug-ins allow developers to access everything they require – from a source standpoint – from within a single interface. The aforementioned features highlight some of the principal ways that these plug-ins assist users with their daily development activities. Enterprises are looking more and more for interconnected solutions to standardize on across an organization, and the WebSphere Studio architecture has allowed for tight integrations to seamlessly incorporate applications.

Resources

- *WebSphere Studio Plug-in Central:* www7b.software.ibm.com/wsdd/downloads/plugin/index.html
- *Ready for IBM WebSphere Studio software validation:* www.developer.ibm.com/websphere/ready.html

Tips for Developers Using an SCM Tool

1. Check in changes often to preserve a development history. This also allows others to integrate your changes.
2. Integrate early and often. Incorporate others' changes to stay current.
3. Mark milestones (checkpoint) via your SCM tool. This also allows you to roll back to a stable state in the future, if needed. Milestones also provide a basis for branching and parallel development.
4. Use SCM to implement process-driven change management. Other things to consider are notifications, integrations, and administration with tools and defect-tracking systems.


Java - Car.java - IBM WebSphere Studio Application Developer
File Edit Source Refactor Navigate Search Project Run Source Integrity Profile Window Help
Check Out
Check In
Add Members
Drop Members
Resynchronize Members
Revert Members
Member Differences
Member History
Member Information
View Sandbox
Source Integrity Preferences
Launch Source Integrity
Package Explorer
CarApplication
src
car
Car.java < v1.6 - mks >
engine
Engine.java < v1.7 > < v1.8 >
engine.block
Piston.java < v1.3 >
Plug.java < v1.3 >
Valve.java < v1.2 >
exhaust
CatalyticConverter.java < v1.3 >
ExhaustSystem.java < v1.2 >
Muffler.java < v1.2 >
suspension
Shock.java < v1.2 >
Spring.java < v1.2 >
transmission
Gear.java < v1.2 - joe >
Transmission.java < v1.5 - joe >
transmission.auto
TorqueConverter.java < v1.4 - joe >
transmission.stick
Clutch.java < v1.2 - joe >
JRE_LIB - C:\Program Files\IBM\WebSphere St
project.pj
Hierarchy Navigator Package Explorer
Tasks (0 items)
Description Resource In Folder Location
engine.block.Piston.java - CarApplication/src

```
ission;
orqueConverter;
Clutch;

ems;

tion.  You can run application and the car will create
 The start method prints "Vroom Vroom" to stdout.
public class Car {

    private static Engine motor;
    private static Transmission tranny;
    private static ExhaustSystem exhaust;
    private static Spring[] springs;
    private static Shock[] shocks;

    private static int milage;

    public static void main(String[] args) {
        milage = 0;
```

FIG. 2: THE JAVA PACKAGES VIEW FROM WSAD SHOWING TEXT FLASH LABEL DECORATORS FROM THE MKS SOURCE INTEGRITY ENTERPRISE EDITION INTEGRATION

www.LinuxWorld.com

Security: Protect Your Linux Environment
LINUXWORLD MAGAZINE
On Location at the Render Farm
DreamWorks is ready to take Linux into new worlds
Linux Alternatives to Microsoft Windows Server Products
Web services and Linux
Embedded Linux with Java

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

$49.99

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

The World's Leading i-Technology Publisher

Use of XSL offers many strategic advantages

Best Practices for Using XSLT in WAS Applications

BY HARVEY **GUNTHER**

XSLT (eXtended Stylesheet Language Transformations) is a very powerful and flexible tool in the XML technology arsenal for transforming XML documents into HTML, plain text, or different XML representations.

ABOUT THE AUTHOR
Harvey Gunther is a senior performance analyst in IBM's WebSphere Portal Server Performance team. Harvey has 16 years of server runtime performance analysis and development background with WebSphere Portal Server, WebSphere Application Server, IMS, VisualAge Smalltalk, and Imaging.

E-MAIL
hgunther@us.ibm.com

The use of XSLT in WebSphere Application Server (WAS) customer applications is burgeoning, despite the following problems:

- XSLT is difficult to learn and hard to master.
- XSLT is not an optimal performer, but its performance is improving rapidly.
- XSLT has little or no tooling for authoring HTML/XHTML pages.

In this article I explore the reasons why some WAS applications use XSL for HTML production instead of JavaServer Pages (JSP). I will compare the performance of XSLT for HTML/XHTML production against JSPs and browser formatting. I will then provide guidance on how to improve XSLT performance in WAS should you decide to go this route. While this article focuses on the use of XSLT for the production of HTML, the performance best practices are directly applicable to other WAS uses of XSLT, such as XML-to-XML transformations and XML-to-text transformations.

Why Some WAS Applications Use XSLT – Even Instead of JSPs

Despite the XSLT obstacles discussed earlier, customers are finding the following advantages over JavaServer Pages when using XSLT for end-user content:

- ***Browser offload:*** Most current Internet browsers can perform XSL transformations. This is a very effective performance offload of end-user content production, and JSPs can't do it. However, as I will discuss later, browser offload is not necessarily a best performance practice that should be used blindly.

- ***Role separation:*** The goal of most applications is to provide complete separation between end-user presentation, or view logic, and business, or model object logic. XSL, with its specialized syntax and self-contained logic, encourages this. In a typical project, the XSL programmers write view logic only, mixing graphic design with end-user presentation. Business application programmers focus on model application logic only. Most application programmers design poor end-user interfaces. JSPs with inline Java scriptlets tend to discourage role separation. On many J2EE application projects, the line between end-user presentation and application logic is badly blurred.

- ***Platform independence:*** JSPs are Java application server artifacts only. XSLT can be a platform-independent universal investment. XSL transformations of XML representations of end-user presentation have much wider applicability. Mainframe applications, Java applications, and other Web production facilities can all reuse the XSL logic that builds HTML/XHTML.

- ***Device independence:*** Application-produced XML can easily be transformed for a wide number of end-user devices, including browsers, pervasive devices, and other programs.

- ***Performance offload:*** Besides browser offload, there is a growing use of hardware acceleration at the network level. The J2EE application replies with an XML response through a proxy. The proxy performs the XML transformation and, in some cases, caches and/or compresses the result. Facilities like this are either home-grown or part of a newly growing breed of network appliance–based hardware accelerators.

Badly Performing WAS Sample Application that Uses XSLT

The rest of this article is based on a sample WAS/J2EE application, *Order View*, which is an order inquiry facility. The application displays an order and its details. An important part of the displayed data is the product family summary of the items that the order contains. The application uses XSL to transform XML representations of the order's objects into an HTML page. The major application steps are as follows:

1. Retrieve all of the Items contained in the Order.
2. Serialize all of the Order Item Java objects into XML. As we will see, this is not an optimal approach.
3. Build an XML model representation of the data to be contained in the HTML page.

4. Transform the XML model into HTML using an XSL stylesheet that groups all of the items into the table of product families. The major feature of the XSL stylesheet is the instruction set that examines each item and builds the product table.

The application produces anywhere from 5KB–814KB worth of XML data, which is transformed to HTML.

Baseline Performance of XSLT for the Sample Application

Figure 1 shows how a WebSphere Application Server (WAS) servlet incorporates and uses XSLT. The figure depicts a *server-side transform*, which applies the XSLT transformation directly in the application, rather than offloading the transformation to the browser or some other process.

Figure 2 compares the performance of using XSLT for server-side transforms as described in Figure 1 with more typical alternatives: JSPs and browser transformation of XML to HTML. For the duration of this article, I will discuss performance best practices for improving the performance of XSLT for server-side transforms. I will also discuss whether or not browser transforms should be considered a best practice. Based on Figure 2, it would seem that sending XML to the browser and allowing the browser to build HTML/XHTML is the winning strategy. However, there are issues – performance and otherwise – concerning browser transformations in WAS applications.

Are XSL Browser Transformations a Best Practice?

Figures 1 and 4 show that sending XML to the browser is the best-performing alternative. Based on Figure 2, this option appears to be more than 10 times better than a server-side transformation of the same stylesheet. Why then shouldn't I declare XSL browser transformations to be *the* best practice and end this article right here? There are several reasons why XSL browser transformations are not the ultimate best practice. The WAS sample application described earlier is reason enough not to do browser XML/XSLT transformations.

- ***Control of the browser population:*** You must have control over your browser population to confidently use XML/XSL browser transformations. While the latest versions of the most prevalent browsers support XML/XSL transformations, are you certain that all of the browsers (or at least 90% of them) that use your application can support XML/XSLT? This is easier to predict or control with most intranet applications. It is largely impossible to control with respect to Internet applications.
- ***Corporate security:*** Sending raw XML to the browser can be a corporate security violation. The raw XML – which is a corporate asset – is available and viewable in the

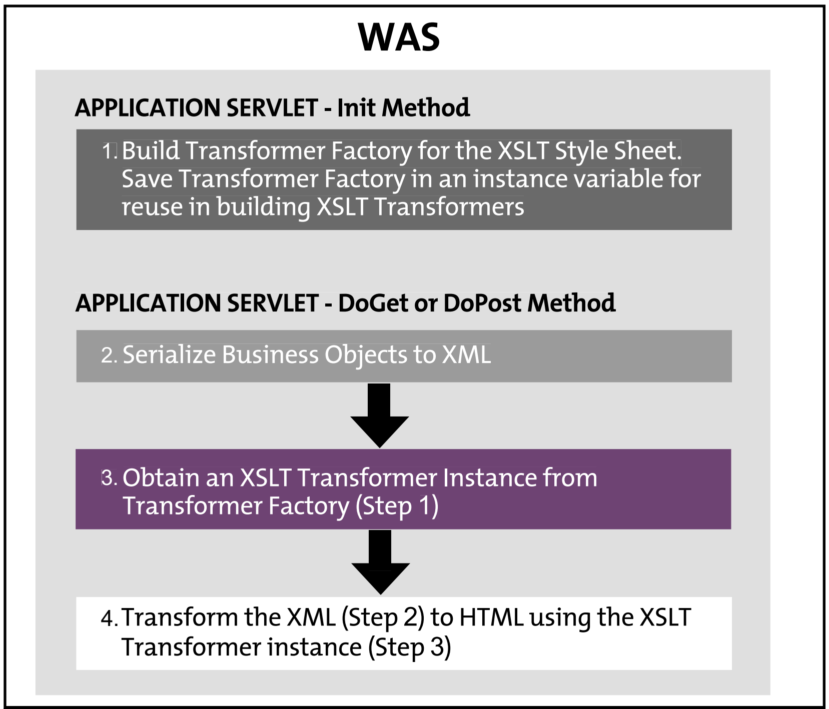

WAS
APPLICATION SERVLET - Init Method
1. Build Transformer Factory for the XSLT Style Sheet. Save Transformer Factory in an instance variable for reuse in building XSLT Transformers
APPLICATION SERVLET - DoGet or DoPost Method
2. Serialize Business Objects to XML
3. Obtain an XSLT Transformer Instance from Transformer Factory (Step 1)
4. Transform the XML (Step 2) to HTML using the XSLT Transformer instance (Step 3)

FIG. 1: CONCEPTUAL USE OF XSLT BY A WAS APPLICATION

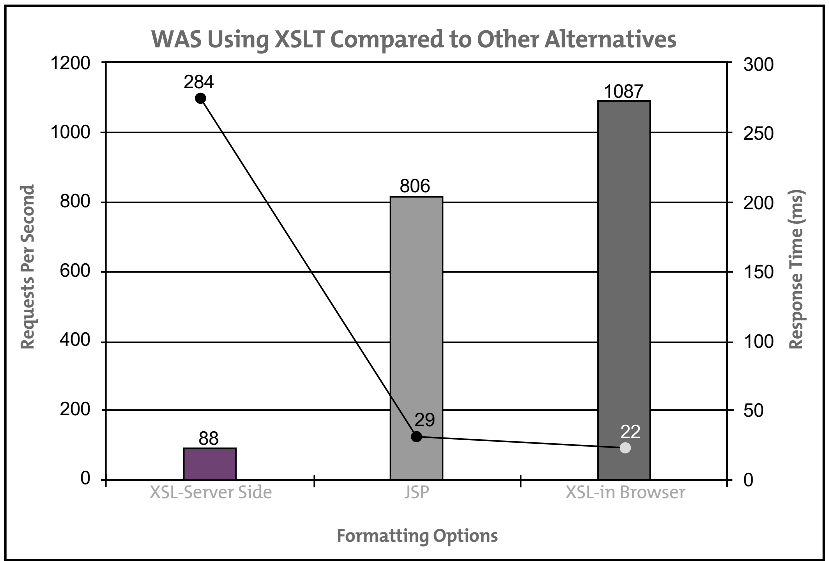


FIG. 2: BASELINE PERFORMANCE OF XSLT COMPARED TO JSPS AND BROWSER OFFLOAD

browser. Depending on the sensitivity and structure of the data, this might constitute a security breach.

- ***Amount of data sent:*** Generally speaking, XML/XSL sent to the browser is smaller than HTML. However, the sample application for this article sends large amounts of XML data to the browser and uses the browser to apply XSL to transform the XML into a smaller and more precise representation. The WAS sample application sends up 814 KB worth of data to the browser. Sending large amounts of XML data to the browser is typically not a problem in lab environments with a high-speed, controlled private network. However, sending large amounts of XML data to the browser over a less-than-perfect network like the Internet can be problematic.
- ***Performance hit in the browser:*** The WAS sample application serializes its business objects and sends them to the browser, which applies XSL to group the objects in a summary view. This kind of business logic in XSL is a bad performer. Sending this to the browser is tantamount to transferring this problem to your end user, who is forced to watch while his browser takes several seconds to format this page.

The Performance Components "Physics" of XSL Transformation

Figure 4, a more detailed look at Figure 1's Step 3, provides a more descriptive examination of the flow of an XSL transformation. Improving performance of the XSLT technology in WAS involves identifying and understanding the performance characteristics and components of that technology. The best practices for performance that I will discuss later are based on this understanding and apply techniques that optimize these performance components.

Here is a better description of the XSLT components:

1. ***Parser:*** The transform engine uses a standard XML parser to parse the input document to build an internal data structure called a DTM. Parser performance is impacted by the size and complexity of nested elements. Reducing the size and complexity in terms of nested elements can optimize this component and improve overall performance. A faster and more optimized parser is an important factor as well.

2. ***Build DTM:*** This component is part of the transformation engine. Like parsing, reducing size and complexity optimizes this component and improves performance.

3. ***Transformer:*** This is the transformation component. It applies the rules and procedures of the XSL stylesheet to produce the desired result. Document size and complexity – and yet more important – stylesheet complexity are the major factors in optimizing this component and overall transformation performance. Using a more optimized transformer engine is an important factor as well.

4. ***Serialization and Output:*** These two components build and present the transformation output. The size and complexity of the transformed output are the major factors.

Now let's examine three specific best practices for XSLTC performance:

1. Use of XSLTC, a compiling version of XSLT
2. Enforcement of the Model-View-Controller (MVC) paradigm in XSLT
3. Use of a custom parser to remove XML serialization and reduce parsing

Refer to Figure 3 to see the measured performance benefit of each of the best practices we will examine. In addition, the analysis of each practice in terms of these XSL performance components is shown in Figure 5.

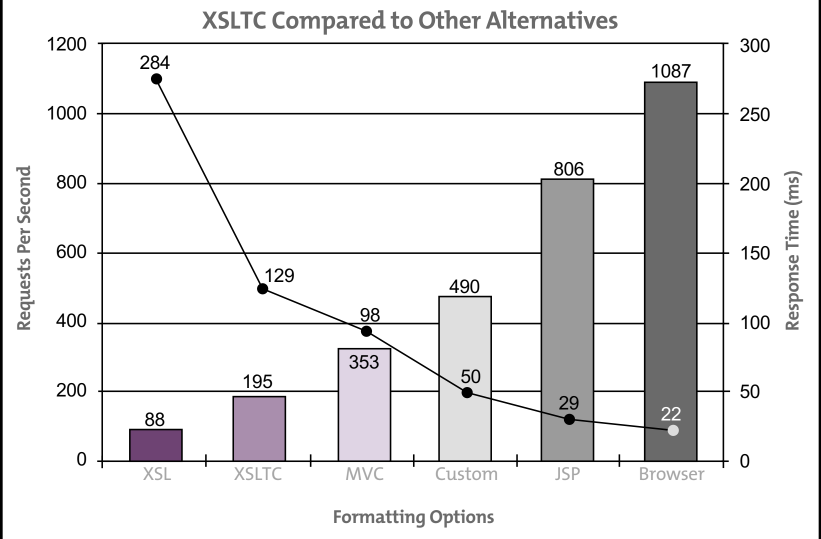
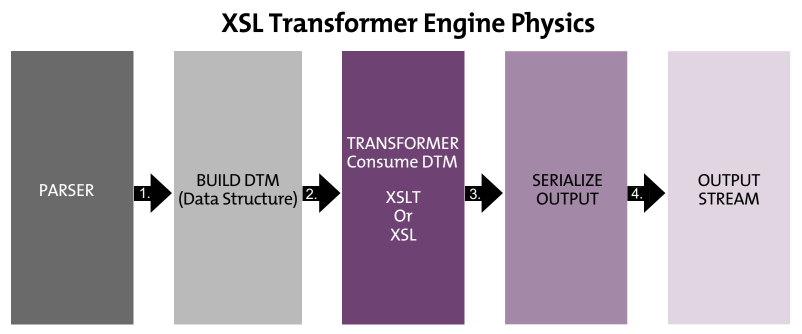


FIG. 3: BEST PRACTICE PERFORMANCE IMPROVEMENT INCREMENTS



XSL Transformer Engine Physics
PARSER
1.
BUILD DTM (Data Structure)
2.
TRANSFORMER Consume DTM XSLT Or XSL
3.
SERIALIZE OUTPUT
4.
OUTPUT STREAM

FIG. 4: XSL TRANSFORMER PERFORMANCE COMPONENTS

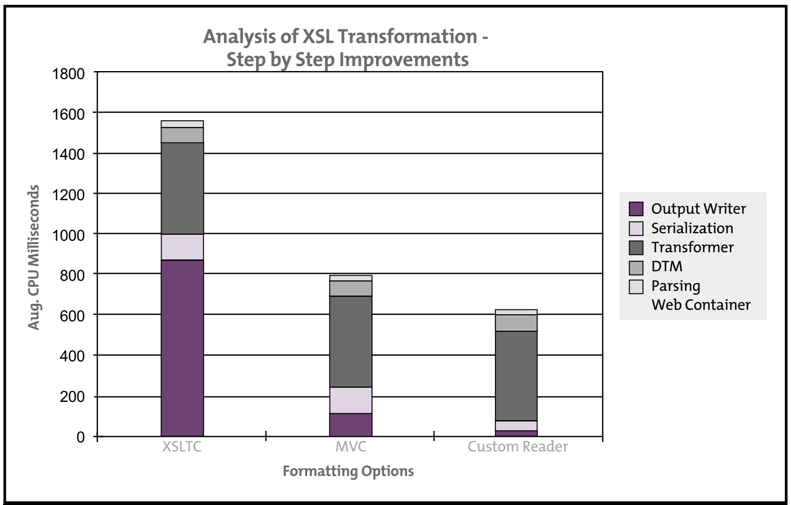


FIG. 5: ANALYSIS OF THE PERFORMANCE INCREMENTS

Best Practices for Using XSLT Transformations

USE COMPILED XSL TRANSFORMATIONS

Apache's Xalan provides two XSL transformers:

1. ***Interpretive:*** The XSL stylesheet is interpreted and built into an XSL processor for each transformation. This is the traditional Xalan XSL transformer, which began its life as Lotus/XSL. This level of XSLT is currently offered in WAS versions 4 and 5. This XSL interpretive exists in two forms: Xalan/C, for C and C++ application functionality; and Xalan/J for Java-based applications. Xalan Interpretive provides maximum flexibility at the expense of speed.

2. ***Compiled:*** XSLTC is a much faster alternative that compiles the stylesheet directly into a Java class. The compile process is conceptually similar to JSP compilation. Applications can compile stylesheets offline and incorporate them with their packaged application. Alternatively, the application can cause the stylesheet to be compiled the first time it is used. In Xalan 2.5.x, available from Apache, XSLTC is now part of xalan.jar, and shares a common code base with Xalan Interpretive. For most stylesheets XSLTC is at least three times faster than the interpretive model.

For the sample application, XSLTC enhances performance by a 3X factor, as is shown in Figure 3. As Figure 5 shows, this best practice improves the transformation XSLTC performance component. Simply put, XSLTC is a much faster transformation engine.

You can obtain XSLTC from Apache and include it with your application for better performance. However, you should be aware that WAS does not currently include XSLTC.

ENFORCE MODEL-VIEW-CONTROLLER WITH XSLT

Model-View-Controller (MVC) is a well-established programming paradigm for separation of responsibility in J2EE Web-based applications. The desired flow is as follows:

1. The servlet request is invoked. (The servlet is the controller.)
2. The servlet accesses business objects to produce dynamic content. (The business objects are the model.)
3. The servlet builds a view object that contains all of the data needed in the response. (The view object is, of course, the view in the MVC paradigm.)
4. The servlet inserts the View object into either the request or the HttpSession, and typically performs a RequestDispatcher .forward to a JavaServer Page or other end-user presentation vehicle.

The sample application serializes business objects to XML and then transforms them into XML. To do this, the application developers were required to insert a fair amount of business logic into the XSL stylesheet. A better approach is to build a view object from the model objects. The application can use the Java business logic to perform the product family grouping in its preparation of this view object.

To enforce the MVC programming paradigm, the application introduces a new object, OrderView, which contains the exact XML representation of the end-user presentation. This change greatly simplifies XSL stylesheet processing and improves performance by 80% (see Figure 3).

Figure 5 shows the analysis behind the performance improvement. Because this best practice reduces the amount of data and complexity, it optimizes the parser and DTM components. However, the simplified transformation rules provide the greater performance relief. The complex product family grouping logic is now performed by the Java server program as it builds the XML representation of the end-user presentation. The reduced XSLT transformation logic amounts to an identity transformation of the OrderView's XML representation.

AVOID XML SERIALIZATION AND PARSING: BUILD AND USE A CUSTOM PARSER

This best practice avoids serializing objects to XML and consequently also greatly reduces parsing overhead. Instead of serializing objects to XML and subsequently parsing with a general-purpose XML parser, develop a custom parser that knows how to produce the same result against the unserialized object.

The custom parser is a Java object that implements all of the same interfaces as a standard JAXP-compliant XML parser but doesn't parse XML. Instead, the custom parser – with exact knowledge of the object – traverses the object's internal state. During this traversal, the custom parser sends SAX events containing the object's data elements to the transformer to build the DTM data structure that is the object of the transformation process.

Listing 1 shows a sample custom parser. Notice that the custom parser has knowledge of the OrderView's state. The handler in Listing 1 also provides the linkage with the XSLT transformation process. As the custom parser traverses the object, it sends SAX events to the transformation handler.

Figure 3 shows a 40% performance improvement, while Figure 5 shows a dramatic improvement in the parsing component shown in Figure 4.

Conclusion

Using XSL for end-user presentation offers many strategic advantages to application developers willing to invest in this ever more important technology.

While use of JSPs offers better performance, with the best practices established in this article – and a good application design – XSL applications can enjoy the strategic advantages that XSL offers, without significantly sacrificing performance.

Acknowledgments

The author acknowledges:

- Gennaro Cuomo, IBM Distinguished Engineer and CTO of the WebSphere Technology Institute
- Ruth Willenborg, manager of the WebSphere Performance team, for her assistance in reviewing this article and securing its publication
- Scott Boag and Joe Latone from Lotus Research
- Henry Zongaro, Sarah McNamara, Morris Kwan, and Brian Minchau from the XSL development team

LISTING 1: CUSTOM PARSER, "ORDERVIEWREADER" FOR THE ORDERVIEW OBJECT

```
public class OrderViewReader implements XMLReader,
ModelReader
{
  public void parse() throws IOException, SAXException
  {
  msHandler.startDocument();
      AttributesImpl envAttrs = new AttributesImpl();
      envAttrs.addAttribute("","","orderdate","NMTOKENS",
      order.getOrderDate());
  msHandler.startElement("","","order",envAttrs);
      msHandler.startElement("","","ordernumber",EMPTY_ATTR);
  msHandler.characters(order.getOrderNumber()
  .toCharArray(),0,order.getOrderNumber().length());
  msHandler.endElement("","","ordernumber");
      msHandler.startElement("","","delivery",EMPTY_ATTR);
  parseAddress(msHandler,order.getDeliveryAddress());
  msHandler.endElement("","","delivery");
      msHandler.startElement("","","billing",EMPTY_ATTR);
  parseAddress(msHandler,order.getBillingAddress());
  msHandler.endElement("","","billing");
      msHandler.startElement("","","product_families",
      EMPTY_ATTR);
      ProductFamilyView[] items = order.getFamilies();
      for(int m = 0; m < items.length; m++)
  {
      parseItem(msHandler,items[m]);
  }
        msHandler.endElement("","","product_families");
        String total_items = String.valueOf(order.
        getTotalItems());
  msHandler.startElement("","","total_items",EMPTY_ATTR);
  msHandler.characters(total_items.toCharArray(),
  0,total_items.length());
  msHandler.endElement("","","total_items");
        String total_value = String.valueOf
        (order.getTotalValue());
  msHandler.startElement("","","total_value",EMPTY_ATTR);
  msHandler.characters(total_value.toCharArray(),
  0,total_value.length());
  msHandler.endElement("","","total_value");
        msHandler.endElement("","","order");
        msHandler.endDocument();
  }
}
```

SYS-CON MEDIA

304,187 of the World's Foremost IT Professionals

DIRECT MAIL, EMAIL OR MULTI-CHANNEL

Target CTOs, CIOs and CXO-level IT professionals and developers who subscribe to SYS-CON Media's industry leading publications

Java Developer's Journal... The leading publication aimed specifically at corporate and independent java development professionals

LinuxWorld Magazine... The premier weekly resource of linux news for executives with key buying influences

Web Services Journal... The only Web Services magazine for CIOs, tech, marketing & product managers, VARs/ISVs, enterprise/app architects & developers

XML Journal... The world's #1 leading XML resource for CEOs, CTOs, technology solution architects, product managers, programmers and developers

PowerBuilder Developer's Journal... The only PowerBuilder resource for corporate and independent enterprise client/server and web developers

WebSphere Developer's Journal... The premier publication for those who design, build, customize, deploy, or administer IBM's WebSphere suite of Web Services software

ColdFusion Developer's Journal... The only publication dedicated to coldfusion web development

WebLogic Developer's Journal... The official magazine for BEA WebLogic application server software developers, IT management & users

.NET Developer's Journal... The must read iTechnology publication for Windows developers & CXO management professionals

Wireless Business & Technology... The wireless magazine for key corporate & engineering managers, and other executives who purchase communications products/services

Recommended for a variety of offers including Java, Internet, enterprise computing, e-business applications, training, software, hardware, data back up and storage, business applications, subscriptions, financial services, high ticket gifts and much more.

NOW AVAILABLE! The SYS-CON Media Database 304,187 postal addresses

e-POSTDIRECT® Your iMarketing Source

For email information: contact Frank at 845-731-3832 frank.cipolla@epostdirect.com
epostdirect.com 800-409-4443 fax845-620-9035

For postal information: contact Kevin at 845-731-2684 kevin.collopy@edithroman.com
edithroman.com 800-223-2194 fax845-620-9035

List Brokerage & Management
EDITH ROMAN
Data Processing • Email Marketing

The better you know your customers, the more you can sell them

Getting a Handle on Your Customers

BY E. KENNETH **NWABUEZE**

Customer online behavior changes all the time. What customers do on your site tomorrow may be different from what they do today. What they searched for while at work may be completely different from what they do at home. Which behavior is the real customer? All of it!

Personalization engines make recommendations based on a visitor's first few clicks. Integrated into the WebSphere content management system is a clickstream analytics platform and a personalization engine. Intelligence generated from the analytics platform is fed into the personalization engine to create profiles for targeting and product recommendation. WebSphere also allows the creation of user-defined profiles with the profile manager and places these into the personalization engine.

Companies can use the analytics platform for basic Web reporting. The WebSphere analytics platform has all the attributes of a basic Web analytics product or service. It also faces the same issue of looking at data in silos. If a company has two distinct domains running on two different WebSphere servers, it runs the risk of not knowing what customers are doing across channels. The WebSphere Application Server can import data from a third-party solution, so an analytics platform can export its analysis into WebSphere.

Just like the analytics platform that comes with WebSphere, most one-size-fits-all products report primarily on page views and visits to the site. If your online revenue is not primarily based on banner advertising, then page views and visits have no value. Instead, you need to generate customized reports that look at your core business model. That means concentrating on your site's revenue-generating properties by defining and tracking key performance indexes that affect your bottom line.

In-depth analysis is not the forte of the WebSphere content management server, so you must find a way to complement its powerful capabilities with other available customer information. While it makes sense to integrate clickstream data with information such as census, demographic, DMA, geo-location, and other data to get a clearer picture of your online customers, that is not what WebSphere was designed to do. Most other Web analytics ASPs can only analyze what they can collect online, lacking the power of a true data-mining platform.

However, there is a third option for enhancing and optimizing the personalization information in WebSphere using your business intelligence platform. Outsourced data-mining companies can easily provide these analytics. One advantage of a full-complement data warehouse and data mining company is that it provides independent data sources. Given the strength of their ETL (extract, transfer, and load) tools, they take just about any data format and produce any type of desired analysis.

A good outsourced company should be designed from the ground up to be a true data warehouse platform. In addition to compiling reports, it will also provide access to detailed data, enabling the creation of custom profiles that are then imported into WebSphere. More important, the outsourced company must be data source independent and possess a flexible ETL tool that can extract and merge separate sources. The company must also allow the user to quickly customize its tools to fit your specific needs.

ABOUT THE AUTHOR
E. Kenneth Nwabueze, founder and CEO of SageMetrics Corporation, has a demonstrated reputation as a developer and entrepreneur of innovative technologies across various high-tech industries. Prior to SageMetrics, he developed business-critical software systems for Buena Vista Pictures and Television. In 2001, President George W. Bush appointed Kenneth to serve on the President's Council of Advisors on Science and Technology.

"Your Web site can offer a wealth of information about your customers and business. Quickly harnessing that knowledge, understanding it, and making it work for you may be the greatest advantage you have over your competitors"

Is it feasible to extract outsourced data from WebSphere? The answer is a resounding yes! If you look at the concept of building a data warehouse, data sources are normally extracted (exported) from the production database and transferred (imported) into a central repository for data mining. Experts recommend that you avoid running processor-intensive applications such as data mining and analysis on a live production database. If the data must be exported to a data repository, then location (outsourced or in-house) becomes a decision based on cost analysis.

Outsourcing the extracted data to a separate system can be done at a fraction of the cost of building a data repository in-house. An outsource provider can determine the best method for extracting and transferring the data from a server. One way is to back up the database at a predetermined frequency and transfer it to repository location. An outsourced provider can use an SCP (Simplified Communication Protocol) to retrieve and securely transfer a copy of the data to their data center. This is not often the preferred method, since the original data source may contain many gigabytes of information. A better solution is an incremental extraction of data, in which only selected tables are exported from the database. This same mechanism can be used to import intelligent and custom profiles into the WebSphere personalization engine.

In a down economy in which customers are hard to come by and acquisition cost is going through the roof, every competitive advantage helps. But just as Amazon.com can grow their customer base faster than any of their store-based competitors, they can lose them just as quickly. Web customers remain loyal only when you provide the best service available. If they can't easily find the products or services they need, if your site is too slow, the process too tedious, or navigation too awkward, they will point their browsers elsewhere.

Your Web site can offer a wealth of information about your customers and business. Quickly harnessing that knowledge, understanding it, and making it work for you may be the greatest advantage you have over your competitors. If you build a good Web site, customers will come. And if you learn how to listen to, collect information on, analyze, and focus on customer needs, they will stay.

WebSphere Journal Advertiser Index...

COMPANY	URL	PHONE	PG
BOWSTREET	WWW.BOWSTREET.COM/EASY	603-559-1900	51
CANDLE CORPORATION	WWW.CANDLE.COM/WWW1/PATHWAI	800-488-4246	2,3
KENETIKS	WWW.KENETIKS.COM	888-KENETIKS	11
MERCURY INTERACTIVE	WWW.MERCURYINTERACTIVE.COM/OPTIMIZEJ2EE		9
MQSOFTWARE	WWW.MQSOFTWARE.COM/QNAMI	800-998-2858	21,25,29
PROLIFICS	WWW.PROLIFICS.COM/WEBSERVICES	800-675-5419	13
QUEST SOFTWARE	HTTP://JAVA.QUEST.COM/JPROBE.WDJ	800-663-4723	31
SYS-CON MEDIA	WWW.SYS-CON.COM/2001/SUB.CFM	888-303-5282	35
SYS-CON MEDIA DATABASE	WWW.EDITHROMAN.COM, WWW.EPOSTDIRECT.COM	800-223-2194, 800-409-4443	45
SYS-CON MEDIA REPRINTS	WWW.SYS-CON.COM	800-802-3026	43
THE LEARNING CONTINUUM COMPANY	WWW.TEACHMEWEBSPHERE.COM	561-953-0095	17
LINUXWORLD MAGAZINE	WWW.SYS-CON.COM	888-303-5282	39
TRILOG GROUP	WWW.FLOWBUILDER.COM	800-818-2199	52
WEBSERVICES EDGE EAST 2004	WWW.SYS-CON.COM	201-802-3069	33
WILY TECHNOLOGY	WWW.WILYTECH.COM	888-GET-WILY	5

WebSphere Journal Coming Next Month...

INTERVIEW
IBM's Rob High, Chief Architect for the WAS Product Family, Discusses the Customer's Influence on Product Architecture
INTERVIEWED BY JACK **MARTIN**

SECURITY
Top Three Security Mistakes Made By WebSphere Developers
BY CALEB **SIMA**

SCALABILITY
HTTP Sessions Within Clustered WebSphere v5 Environments
BY CHRIS **DELGADO**

MIGRATION
Rules for Notes/Domino Developers Moving to WebSphere
BY MARK **DIXON**

PARASOFT RELEASES JTEST 5.0, AUTOMATED ERROR PREVENTION

(Monrovia, CA) – Parasoft, a provider of Automated Error Prevention software solutions, has announced the general availability of Jtest 5.0, which automates all aspects of Java unit testing and coding-standards compliance. This latest version is equipped with new JUnit test-generation capabilities, automated code correction capabilities, and other features designed to simplify team-wide Java error prevention.

When there's no margin for error...

Based on the Parasoft AEP Methodology, Jtest is designed to help developers catch errors early in development and prevent entire classes of similar errors in the future by automating practices such as unit testing and coding-standard checking.

"Automated error prevention is most effective when implemented within a team environment, says Gary Brunell, Parasoft vice president of Professional Services. "We've enhanced the team support so that development groups can efficiently work together to produce reliable software. We also completely redesigned the user interface to make testing as fast and easy as possible, and to seamlessly integrate into IBM WebSphere Studio Application Developer and Eclipse IDEs."

New features of Jtest 5.0 include:

- JUnit format unit-test cases are automatically generated and executed for instant verification and white-box testing. Jtest also runs any valid JUnit test cases, monitors their coverage, and uses them for automated regression testing.
- The Quick Fix feature automatically corrects violations of over 160 coding standard rules and adds Design by Contract comments to the code.
- Team Server manages team-wide sharing of test settings and files.

www.parasoft.com

VISUAL SLICKEDIT PLUG-IN EARNS READY FOR WEBSPHERE VALIDATION

(Morrisville, NC) – SlickEdit Inc., provider of a comprehensive and flexible code editor for software developers, has announced that Visual SlickEdit Plug-In for WebSphere Studio & Eclipse has been validated by IBM as Ready for WebSphere Studio software. Visual SlickEdit Plug-In is a feature-rich, highly customizable code editor that supports a wide range of programming languages. It provides advanced editing features and capabilities to help developers code faster. The Ready for WebSphere Studio software validation assures customers that WebSphere plug-ins meet the highest standards of WebSphere Studio integration.

Visual SlickEdit Plug-In for WebSphere Studio & Eclipse is an advanced code editor that plugs right into both WebSphere Studio and Eclipse integrated development environments. WebSphere Studio is built on the open-source Eclipse framework and is designed to allow plug-ins to extend and add value to it.

www.slickedit.com

VIACORE COMMUNITY KICKSTART ENABLES QUICK COMMUNITY INTEGRATION DEPLOYMENT

(Irvine, CA) – Viacore, an integration utility for private trading communities, has announced the availability of its Community KickStart, a set of pre-packaged, fixed-price services that can enable – in as few as 90 days – community integration deployment when used in conjunction with IBM WebSphere Business Integration Connect. Viacore Community KickStart is ideal for companies looking to get integrated trading communities up and running quickly.

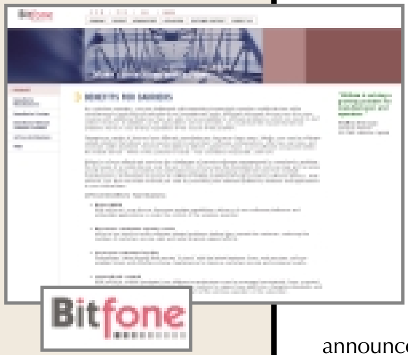
This offering marks Viacore's first milestone in their recently announced agreement with IBM in which Viacore's BusinessTone com-

Bitfone to Extend WebSphere Device Management Software with Over-the-Air Updates

(Las Vegas) – Bitfone Corporation, a provider of over-the-air (OTA) firmware update technology for mobile phones, has announced it will integrate its mProve solution into IBM's WebSphere device management software. The combined solution is aimed at helping service providers more efficiently update software on devices remotely – an increasingly important feature as the use of wireless devices rises. The two companies will jointly market the solution. WebSphere device management software incorporates IBM Tivoli technology.

"Solutions such as Bitfone's mProve provide a convenient way for service providers to remotely resolve mobile phone software problems," says Jonathan Prial, vice president, Sales and Business Development for IBM. "The integration of WebSphere device management software with the mProve solution can help customers to avoid software-related recalls and reduce customer support costs, thus helping service providers deliver better and more cost-effective customer service in an increasingly competitive wireless landscape."

"OTA firmware update is a leading-edge technology, and working with a company like IBM will help to accelerate adoption by the leading wireless operators around the world," says Gene Wang, chairman and CEO of Bitfone. "mProve enhances the value proposition of device management software such as IBM's by providing the mechanism to update the core software on mobile phones down-to-the-metal."



www.bitfone.com

munity integration services will be available through IBM in conjunction with IBM's WebSphere Business Integration Connect software.

"Viacore Community KickStart can help accelerate the deployment of WebSphere Business Integration Connect communities by assessing how ready participants are to join an integrated community, testing a participant's connectivity capabilities, and managing the activation process for participants," says Chris Gift, director of marketing for Viacore. "Our goal is to provide community builders with a pre-packaged baseline set of highly automated services that can help reduce the time and cost of implementing their B2B initiatives, while at the same time enabling their initial set of trading partners to become fully functional participants in as few as 90 days."
www.viacore.net

NEON'S J2EE-COMPLIANT SHADOW ADAPTERS RECEIVE SELF-TEST FOR WAS V5.0

(Sugar Land, TX) – NEON Systems, Inc., a provider of enterprise-class mainframe integration, has announced that its J2EE–compliant Shadow adapters have received Self-Test approval for IBM WebSphere Application Server V5.0. WebSphere Self-Testing is a program that facilitates the self-testing of WebSphere complementary technologies through an IBM-endorsed testing process.

NEON's Shadow mainframe integration technology simplifies access to mainframe data sources when integrating application platform suites such as the WebSphere platform with IBM eServers running OS/390 or z/OS operating systems. Shadow technology, in combination with WebSphere, helps to close the gap between business strategy and information technology, allowing joint customers to create and operate a dynamic e-business by providing the most functional and robust solution for application-platform-suite-to-mainframe integration.

"We have worked with IBM on our Shadow mainframe adapter technology from its inception, delivering both direct sales and OEM versions of the product," said Jonathan Reed, NEON's director of Channel Sales. "Shadow makes the vast wealth of mainframe assets (data, business logic, and applications) available to the industry-leading WebSphere application platform suite, allowing simple standards-based development of new high-value applications. Successful completion of the WebSphere Self-Test (jointly developed by IBM and NEON) demonstrates our commitment to the highest standards of performance and interoperability."

Shadow's industry-standard mainframe adapters enable WebSphere developers to exploit the fastest available transactional interface to a particular data or application target – all within an environment that is fully instrumented for systems management – across a broad array of mainframe resources.
www.neonsys.com

Citrix Enhances Integration Between WebSphere Portal and MetaFrame Presentation Server

(Orlando, FL) – At Citrix iForum 2003, Citrix Systems, Inc., a provider of access infrastructure solutions, announced the release of a new portlet that provides integration with IBM WebSphere Portal, giving joint customers secure, easy, and instant access – on demand – from their WebSphere Portal environment to heterogeneous applications and information resources centrally managed by Citrix MetaFrame Presentation Server.

Citrix and IBM have worked over the past decade to bring integrated enterprise access solutions to customers across a host of industries. The announcement reaffirms Citrix's commitment to giving customers what they need to transform their businesses into productive and secure on-demand enterprises.

An earlier release of the portlet, introduced last year, is still one of the top five most downloaded portlets in the IBM WebSphere catalog. The updated portlet, designed for Citrix customers and developed using the latest releases of IBM WebSphere and MetaFrame Presentation Server, gives customers secure, single-point access to Web applications, content, and resources, as well as custom or commercially packaged applications, without the need for application rewrites or reprogramming.

The new portlet includes multiple authentication modes, increasing both resource and application security – key issues for enterprise customers. In addition, the new portlet provides Java client support; configurable user settings for sound, color and window size; and built-in multifarm support. The portlet is built on a flexible architecture so customers can customize the solution to meet their individual scalability and fault-tolerance requirements.

"With WebSphere Portal, Citrix customers can broaden the use of their WebSphere environments with secure access to a wide variety of applications and can accelerate their return on investment by improving employee productivity and customer service," says Tim Thatcher, program director of WebSphere Portal.

Citrix MetaFrame Presentation Server is one of the world's most widely deployed presentation servers for centrally managing heterogeneous applications and delivering their functionality as a service to workers, wherever they may be, according to the company.
www.citrix.com

RSS Feed Now Available on WJ Home Page

The *WJ* home page (www.sys-con.com/websphere) now offers an RSS feed (www.sys-con.com/WebSphere/feed.rss) to help you keep current with the latest goings-on in the WebSphere community. Sign up today to start receiving timely updates of product announcements, news items, and other important happenings in the world of WebSphere.

On Protecting U.S. IT Jobs

BY ADAM **KOLAWA**

As a closing thought, let's consider our jobs and how to protect them. There is a lot of instability in today's software industry, and the American sector is losing its competitive edge. The problem is that U.S. developers are overpaid with respect to what we produce, and people in other countries are willing to produce a similar product for significantly less money.

We can worry and complain about this, but that won't eliminate the threat. In just seconds our source code can be sent overseas, where it can be completed or fixed by other developers. If these overseas developers are smart – and they are – they will be able to read the code and start working on it immediately. There is no need for us any more. This is the sad truth.

So, what can we do? I don't think anyone wants to lower their standard of living, so taking a tremendous pay cut isn't an option. In that case, the only real solution is to determine how to become more competitive and improve our productivity. In other words, we need to find ways to increase our output.

There are many ways the software development process could be improved. To increase productivity as quickly as possible, I suggest we focus on a low-hanging fruit that can deliver large returns: we should shift the distribution of time spent building new code versus debugging. On average, we spend only 20% of our time writing new code and 80% of our time chasing and eliminating bugs. If we could only shift this balance so that we spent 40% of our time developing and 60% debugging, we would be 100% more productive, 50% less expensive, and significantly more competitive.

"If we could only shift this balance so that we spent 40% of our time developing and 60% debugging, we would be 100% more productive, 50% less expensive, and significantly more competitive"

If we really want to achieve such gains, we need to introduce fewer bugs into our code so we don't waste so much time finding and fixing them. This general concept is called error prevention; when it's automated, it's called Automated Error Prevention.

The basic idea is that you should learn from your mistakes. Each time you identify an error, you want to determine why it occurred, create an error prevention practice to prevent similar types of errors from occurring, and then automate the error prevention practice that you've developed.

To fully implement Automated Error Prevention, you should perform this process for each error that you find. However, if you want immediate results, you need to take some shortcuts. The best shortcuts I know of are coding standards and unit testing, two practices that are easy to implement and that can be completely automated.

Coding standards and unit testing are valuable shortcuts because they can significantly reduce the error rate. In the case of coding standards, industry experts have already done the groundwork. They examined the most common and serious errors that developers tend to introduce, and then created rules we can follow to avoid these errors. The same is true for unit testing. Automated white-box unit testing can flush a lot of uncaught runtime exceptions, so we don't need to find them ourselves.

If we could also automate the process of fixing these problems, we could be very productive. In fact, you can already do this if you develop with Java. Tools exist that can automatically identify and correct Java code problems so that Automated Error Prevention can be implemented as rapidly and efficiently as possible.

Automating coding standards and unit testing is a good start to improving our productivity, but we will have to do much more to ensure our job security in these unstable times. For many years, we in the software industry have been very arrogant. Many of us now believe that the way we build things is the only possible way, and that our brains are irreplaceable. People in other industries believed this once – until their jobs were lost to competitors who could build the same product more efficiently.

Sadly, we too have been proven wrong, and now our jobs are on the line. In the good old days, our managers begged us to use tools to make us more productive, and we refused. Now we will have to beg them to provide us with tools that make us more productive – in the hope that they will save us from being replaced. Quite scary, isn't it?

ABOUT THE AUTHOR... Adam Kolawa, PhD, is chairman/CEO of Parasoft, a company that provides Automated Error Prevention solutions that combine advanced products, services and expertise to help companies automatically prevent errors throughout the software life cycle. Parasoft recently released Jtest 5.0, the first development product of its type to automate all aspects of Java unit testing and coding standards compliance.

E-MAIL
sk@parasoft.com

Build WebSphere Portlets FAST*

**and change them even faster.*

Out-of-the-box portlets aren't enough.
But, you already know that.

Driving directions, stock quotes, and news are nice, but they're not what you need to run your business. What you need are customizable portlets that bring your enterprise IT assets to your enterprise portals.

Bowstreet™ Portlet Factory is the first product to automate the rapid development, deployment, and maintenance of highly adaptive portlets that plug right into the IBM WebSphere Portal stack.

BOWSTREET PORTLET FACTORY
TRY IT FREE FOR 60 DAYS.
WWW.BOWSTREET.COM/EASY

bowstreet

J2EE IS HARD. WE MAKE IT EASY.

XSP™

makes

J2EE

and

XML

deliver

A.S.A.P.

Visit our

WWW.

TRILOG GROUP®

Transfer Your Domino Development Investment Seamlessly to J2EE RAD Development Using XSP™!

If you're a Domino shop or Domino developer and you want to enter the world of J2EE, FlowBuilder® XML Edition is just what you need. FlowBuilder's Visual XSP Studio provides standards-based ultra-rapid application development with a high-level component framework, seamlessly integrated workflow engine and organization modeling. Plus, it offers universal XML data connectivity, transactional and document-based development and integration with web services. Oh, and did we mention your Domino skills and knowledge are completely transferable? So check out FlowBuilder® A.S.A.P. at www.flowbuilder.com.

800-818-2199/781-937-9963

www.flowbuilder.com

sales@triloggroup.com

FlowBuilder

For people in the know.