

# WebSphere®

## DEVELOPER'S JOURNAL

The World's Leading Independent  
WebSphere Developer Resource

WebSphereDevelopersJournal.com

JUNE VOLUME 2 ISSUE 6

**SAVE UP TO \$400**

see page 31 for details



**SUBSCRIBE TODAY**

and get UP TO **3 FREE CDs!**  
(WHILE SUPPLIES LAST)

FROM THE EDITOR  
**On-Demand Computing  
Could Ignite Industry**

BY JACK MARTIN • PAGE 4

BOOK EXCERPT  
**Building J2EE Applications  
with IBM WebSphere**

PAGE 28

PRODUCT REVIEW

**M7 Application  
Assembly Suite**

BY JAY JOHNSON • PAGE 42

FINAL THOUGHTS

**Improving the  
Decision-Making Process**

BY SUMIT DESHPANDE • PAGE 50

DISPLAY UNTIL AUGUST 31, 2003

\$8.99US \$9.99CAN



0 09281 03422 3

**SYS.CON  
MEDIA**



BY ZHE WANG AND QI LI PAGE 20

**Implementing WebSphere  
Security Through LDAP**

*Part 2: Setting up LDAP authentication*

BY LOU MAUGET

PAGE 8

**WebSphere and Database  
Performance Tuning**

*Part 2: Application development and monitoring*

BY MICHAEL S. PALLOS

PAGE 16

**Manage Your Applications**

*View management of HTTP and EJB requests*

BY MICHAEL J. MORTON

PAGE 32

**Step-By-Step EJB 2.0  
CMP/CMR Development**

*Redefined model offers better performance and flexibility*

BY SWARAJ PAL ET AL.

PAGE 38

**Accelerating Delivery of Personalized Content**

*Enjoy the same advantages as static-content publishers*

BY STEVE IMS

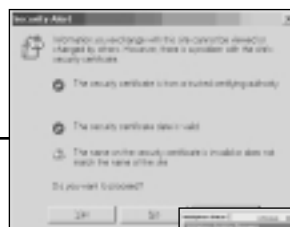
PAGE 44

**Web Services Invocation Framework**

*Part 1: WSIF architecture*

BY BORIS LUBLINSKY

PAGE 46



# PROLIFICS

[WWW.PROLIFICS.COM/WEBSERVICES](http://WWW.PROLIFICS.COM/WEBSERVICES)

# WILY TECHNOLOGY

[WWW.WILYTECH.COM](http://WWW.WILYTECH.COM)

# On-Demand Computing Could Ignite Industry

BY JACK MARTIN

The Internet has changed the way businesses operate. With the ability to purchase from anyone, regardless of location, buyers today know more about products, pricing, and availability than ever before. This shifts more power to buyers by helping them get the best price and terms in nearly every case.

IBM is putting a very heavy emphasis on the concept of on-demand computing for the enterprise. WebSphere is the key software platform and makes up the integral components of an on-demand operating environment for IBM's on-demand strategy.

The promise of the on-demand operating environment is to integrate the people, processes, and information throughout the entire enterprise. It is intended to virtualize IT resources to improve utilization and align IT expenses with business needs while leveraging automated technologies to manage IT resources.

The key thing to remember is that none of this really works yet.

No one has ever run a shoe shine stand with a computer that automatically heals itself and with software that is constantly morphing around the business's current needs, endlessly anticipating its future needs while simultaneously going out on the Internet and taking reservations for shoe shines and offering special customers an opportunity to purchase a virtual prepaid shoe shine card. There has never been a computer that constantly monitors the current price of shoe polish and polishing rags so the stand operator will know to buy in bulk when prices are favorable and to buy supplies on an as-needed basis when it is economically in his favor. There is no computer that is always ready to identify new trends in shoe colors so the stand operator can capitalize on new color trends ahead of other stand operators.

There is no computer monitoring how global political and economic variables like oil prices or interest rates could affect demand for a shoe shine or that is alert to the possibility of hackers making off with all of the stand's customer credit card accounts or that enables the stand operator to see and manage the shoe shine stand as an integrated whole, even if other



companies handle important processes like cleaning the polishing rags.


If there were such a computer, the only thing left for the stand operator to worry about would be if someone were to come along and knock over the telephone pole across the street, resulting in a loss of connectivity to the Internet – if the owner doesn't have a wireless

backup, just in case.

The fact that on-demand computing isn't a reality yet is the bad news. The good news is that if corporate America salutes this concept – and there are a few corporations out there who have – this will open an unprecedented group of opportunities for every software and hardware vendor in the world, now and for the next generation. This will make the dotcom bubble look like a momentary lapse of reason, nothing more, nothing less.

Just think of it, customer requirements demanding wholesale integration of applications, processes, and content across and beyond their enterprise, with the ability to expand access to computing resources on demand and the dynamic construction and assembly of new business services that leverage existing business systems. It will take two or three generations of software on a global basis for everyone to just agree on how these deliverables will work and the metrics for measuring them.

Countless new products will be born in the rush to fulfill the promise of on-demand technology. Many new billionaires will be minted – maybe you will be one of them. For this time it will not be just the technologists who will be in a position to experience a windfall, but also the business people who figure out how to get a competitive edge by using on-demand technology in their businesses. We may be entering the final phase of information technology, in which the machines start to run our businesses, and ultimately, our lives.

This may make some people a little crazy, but it's progress. More important, on-demand computing will once again give everyone's sales staff something to sell that has sizzle. 

**ABOUT THE AUTHOR...** Jack Martin, editor-in-chief of *WebSphere Developer's 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. **E-MAIL...** jack@sys-con.com

### 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  
EDITORIAL DIRECTOR JEREMY GEELAN  
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 STILLER

### WRITERS IN THIS ISSUE

Sumit Deshpande, Jay Johnson, Qi Li, Boris Lublinsky, Jack Martin, Lou Mauget, Michael Morton, Raju Mukherjee, Swaraj Pal, Michael S. Pallos, Deependra Shrestha, Raghu Shrestha, Zhe Wang

### SUBSCRIPTIONS

For subscriptions and requests for bulk orders, please send your letters to Subscription Department. [SUBSCRIBE@SYS-CON.COM](mailto: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 A. KIRCAALI  
SENIOR VP, SALES & MARKETING CARMEN GONZALEZ  
VP, INFORMATION SYSTEMS ROBERT DIAMOND  
VP, SALES & MARKETING MILES SILVERMAN  
PRESIDENT, SYS-CON EVENTS GRISHA DAVIDA  
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  
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 SHELLA DICKERSON  
CUSTOMER RELATIONS MANAGER RACHEL MCCOURAN

### 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](mailto:SUBSCRIBE@SYS-CON.COM)  
WebSphere® Developer's Journal (ISSN# 1535-6914)  
is published monthly (12 times a year).  
Postmaster send address changes to:  
WebSphere Developer's Journal, SYS-CON Publications, Inc.  
135 Chestnut Ridge Road, Montvale, NJ 07645

© COPYRIGHT

2003 BY SYS-CON PUBLICATIONS, INC. ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY OR ANY INFORMATION STORAGE AND RETRIEVAL SYSTEM, WITHOUT WRITTEN PERMISSION. FOR PROMOTIONAL REPORTS, CONTACT REPORT COORDINATOR. SYS-CON PUBLICATIONS, INC. RESERVES THE RIGHT TO REUSE, REPRODUCE AND AUTHORIZE THE READERS TO USE THE ARTICLES SUBMITTED FOR PUBLICATION.

ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES ARE TRADE NAMES. SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES. SYS-CON PUBLICATIONS, INC. IS NOT AFFILIATED WITH THE COMPANIES OR PRODUCTS COVERED IN WEBSHERE DEVELOPER'S JOURNAL.

WEBSHERE® IS A REGISTERED TRADEMARK OF IBM CORP. AND IS USED BY SYS-CON MEDIA UNDER LICENSE. WEBSHERE® DEVELOPER'S JOURNAL IS PUBLISHED INDEPENDENTLY OF IBM CORP., WHICH IS NOT RESPONSIBLE IN ANY WAY FOR ITS CONTENT.

**SYS-CON**  
MEDIA

# PRECISE

[WWW.PRECISE.COM/WSDJ](http://WWW.PRECISE.COM/WSDJ)

# MQSOFTWARE

[WWW.MQSOFTWARE.COM](http://WWW.MQSOFTWARE.COM)

# MQSOFTWARE

[WWW.MQSOFTWARE.COM](http://WWW.MQSOFTWARE.COM)

*Part 2: Setting up LDAP authentication*

# Implementing WebSphere Security Through LDAP

BY LOU MAUGET

The use of Lightweight Directory Authentication Protocol (LDAP) for WebSphere authentication and authorization offers the advantages of single sign-on across application servers and a vendor-neutral protocol and API.

**ABOUT THE AUTHOR**

Lou Mauget is a senior consultant with CrossLogic Corporation, where his assignments include J2EE mentoring and designing instructional material. He has coauthored three computer-related books. His prior career at IBM included work as a consultant, programmer, and developer.

**E-MAIL**

lmauget@crosslogic.com

**P**art 1 of this two-part series showed how to set up a directory and sample application infrastructure for demonstrating WebSphere authentication using LDAP. In Part 2 I take you through the process of setting up LDAP authentication for WebSphere through the following tasks:

- Defining roles and constraints in the Web application deployment descriptor
- Mapping roles to LDAP group entries in the enterprise application deployment descriptor
- Configuring WebSphere to use an LDAP principal registry
- Demonstrating the secured application features enabled by WebSphere and LDAP
- Locking down the LDAP and browser transports using SSL (Secure Sockets Layer)

**Defining Web Application Roles and Constraints**

In this section I will show you how to define security roles and the constraints associated with them. Note that the roles and constraints will have no effect until security for the application server is enabled.

**SECURITY ROLES**

Carry out the following steps to create user and admin roles:

1. Open SecurityDemoWeb/Web Deployment Descriptor from the J2EE Navigator view.
2. Click the Security tab.
3. The top Security Roles tab should be selected. Press the Add button. A new role appears in the Security Roles box.
4. Edit the role in place to read "user".
5. Supply a description of End-user.
6. Add another role named admin.
7. Supply a description of Administrator.

**USER SECURITY CONSTRAINT**

Now we will relate Web resources to roles and transmission integrity. The following steps associate constraints with roles:

1. Click the Security Constraints tab on the deployment descriptor Security page.
2. Click the Add button to create a new security constraint. The constraint appears. In addition, a new Web resource collection appears in the right-hand group.
3. Press the Edit button for the Web resource collection. The Web

Resource Collection dialog appears.

4. Name the collection user-collection.
5. All HTTP methods are constrained if you select none. Leave them unselected.
6. Press the Add button to add a URL pattern.
7. Change the pattern to "/\*". This means all URLs in your document root are constrained by this constraint.
8. Press OK to update the Web resource collection.
9. Press the Edit button in the Authorized Roles group. The Define Authorization Constraint dialog appears. This is where you associate a role with a constraint.
10. Enter a description of the User constraint.
11. Select Role Name user (see Figure 1).
12. Press OK.

**ADMIN SECURITY CONSTRAINT**

Carry out the following series of steps to create an administration security constraint:

1. In the Security Constraints group press the Add button.
2. Press the Edit button for the Web resource collection. The Web Resource Collection dialog appears.
3. Name the collection "admin-collection".
4. All HTTP methods will be constrained if you do not specify any of them. Leave all of them unselected.
5. Press the Add button to add a URL pattern.
6. This time, change the pattern to "/admin.jsp". This means that the given URL is constrained by this constraint. It is also constrained by the previous wildcard constraint. This means all administrators must also be users, but not the converse.
7. Press OK to update the Web resource collection.



8. Press the Edit button in the Authorized Roles group. The Define Authorization Constraint dialog appears.
9. Enter a description of the Administrator constraint.
10. Select Role Name admin (see Figure 2).
13. Press OK.

#### AUTHENTICATION METHOD

Use the following steps to specify the authentication method for this application:

1. Select the Pages tab of the Web deployment descriptor.
2. In the Login group set the Authentication method dropdown to Basic. This means that the browser will pop up a login window when it receives an HTTP status of 401, meaning not authorized. WebSphere will receive the login credentials and try to authorize the user.
3. Save and close the Web deployment descriptor.

### Map Roles to LDAP Group Names

So far the two security roles have no teeth in them. They need to be mapped to names in a user registry. WebSphere needs to map the role user to the LDAP users entry, and role admin to the LDAP admins entry. Carry out this mapping in the SecurityDemo enterprise application deployment descriptor with the following steps:

1. Open the SecurityDemo application deployment descriptor from the J2EE Navigator.
2. Press the Security tab.
3. Set WebSphere Bindings to Users/Groups.
4. Use the Add button in the Security group to add the role named user.
5. Use the Add button in the Groups group to add the LDAP group named users.
6. Use the Add button in the Security group to add the role named admin.
7. Use the Add button in the Groups group to add the LDAP group named admins.
8. Save and close the deployment descriptor.

### Application Server

The easy way to configure the application server is to attempt to run the application on it. Afterward you can enable security on the new server configuration.

#### CONFIGURE THE APPLICATION SERVER

WebSphere Studio includes a WebSphere V5.0 Test Environment server that is an actual WebSphere application server. Follow these steps to create a server configuration for WebSphere Studio:

1. In the J2EE Navigator view, right-click project SecurityDemoWeb. Choose Run on Server. The Server Selection dialog appears.
2. Click the option to Set server as the project default (do not prompt). The dialog should resemble that shown in Figure 3.
3. Press OK to create the server and run the application. The start page should appear in the browser.

#### WEBSPHERE SECURITY

Notice that you were not asked for credentials and that you can access all three pages. This is because server security is enabled. Now you will need to enable security using the new LDAP user registry for authentication and role mapping.

### Administration Console

The configuration page in WebSphere Studio does not expose the information needed to configure LTPA to use your LDAP user registry, so you need to use the WebSphere administration console. Follow these steps to configure LTPA:

1. Open the Server Perspective.
2. In the Server Configuration view, open the server configuration by right-clicking Servers|WebSphere V5.0 Test Environment and choosing Open.
3. Click the Configuration tab.
4. Set Enable administration console.
5. Clear Enable universal test client.
6. Save the configuration and close it.
7. In the Servers view, right-click the WebSphere V5.0 Test Environment. Choose Restart.
8. Wait for the console message.

“Server server1 for eBusiness,” signifying that restart is complete.

9. Open an external browser to <http://localhost:9090/admin>.
10. The console appears as shown in Figure 4. Note that there is no password field because security is not enabled.
11. Enter your last name as the User ID and press OK. The Administrative Console will open.

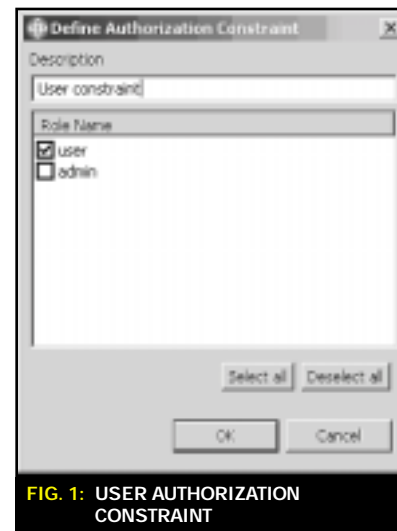


FIG. 1: USER AUTHORIZATION CONSTRAINT

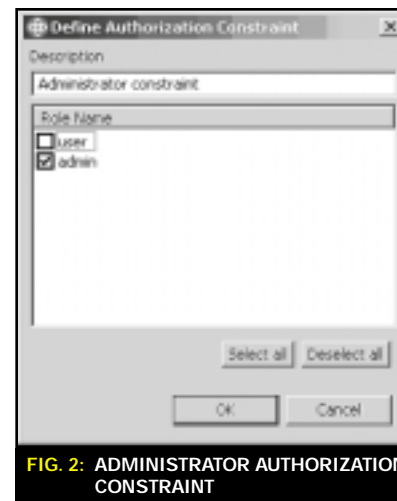


FIG. 2: ADMINISTRATOR AUTHORIZATION CONSTRAINT

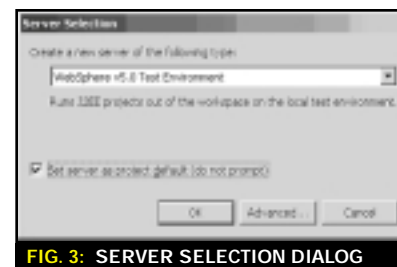


FIG. 3: SERVER SELECTION DIALOG

### Configure Settings for an LDAP User Registry

Ensure that the IBM Directory Server 5.1 is running and then carry out the following steps:

1. Expand the Security|User Registries node in the left-hand pane.
2. In the Server User ID field, enter “was”. Remember that this is the uid attribute value of the DN value cn=was,ou=people,o=rogers60,c=us in your LDAP directory.
3. In the Server User Password field, enter the value “secret”. This is the userPassword attribute value for DN value cn=was,ou=people,o=rogers60,c=us. Note that this

value is invisible to nonprivileged users of the directory.

4. From the Type dropdown list select IBM\_Directory\_Server.
5. Enter a host name of localhost or your remote host DNS name or IP address.
6. In the Base Distinguished Name (DN) field, enter your suffix, “o=rogers60,c=us”.
7. In the Bind Distinguished Name (DN) field, enter “cn=root”. This is the administration user ID for the LDAP directory that you defined previously.
8. In the Bind Password field, enter the value “secret”.
9. Set the Ignore Case option.
10. Click the Apply button at the bottom.
11. A message invites you to press Save to apply the changes to the master configuration (see Figure 5).
12. Press Save on the tab bar or in the Messages pane. A confirmation message will appear.
13. Save to Master Configuration.

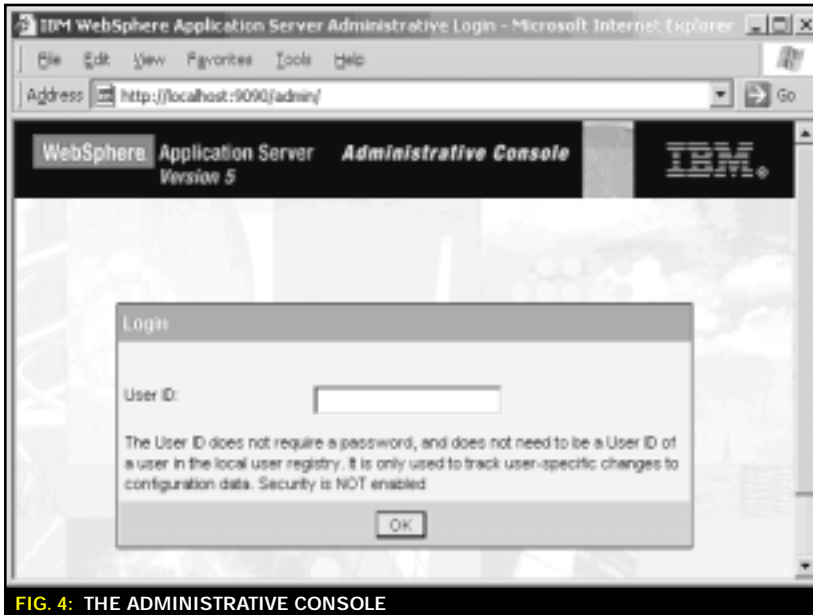


FIG. 4: THE ADMINISTRATIVE CONSOLE

### Configure LTPA

The WebSphere server will use LTPA for authentication. This is a forwardable security scheme that makes single sign-on (SSO) possible across applications running in participating Lotus Domino servers and WebSphere servers. Carry out the following steps:

1. Click the Security|LTPA tree node.
2. In the Password field enter the value “secret”.
3. In the Confirm Password field enter the value “secret”.
4. Press the tab labeled Generate Keys.
5. When the Message(s) box appears, press Save to apply changes.
6. Save to Master Configuration.

### Enable Global Security

You have configured security settings. It is time to turn them on in the server. Use the following steps:

1. Click the Security|Global Security tree node.
2. Set the Enabled option.
3. The Enforce Java 2 Security option is set automatically. Leave it set.
4. Set the Active Security Mechanism dropdown to LTPA.
5. Set the Active User Registry drop down to LDAP.

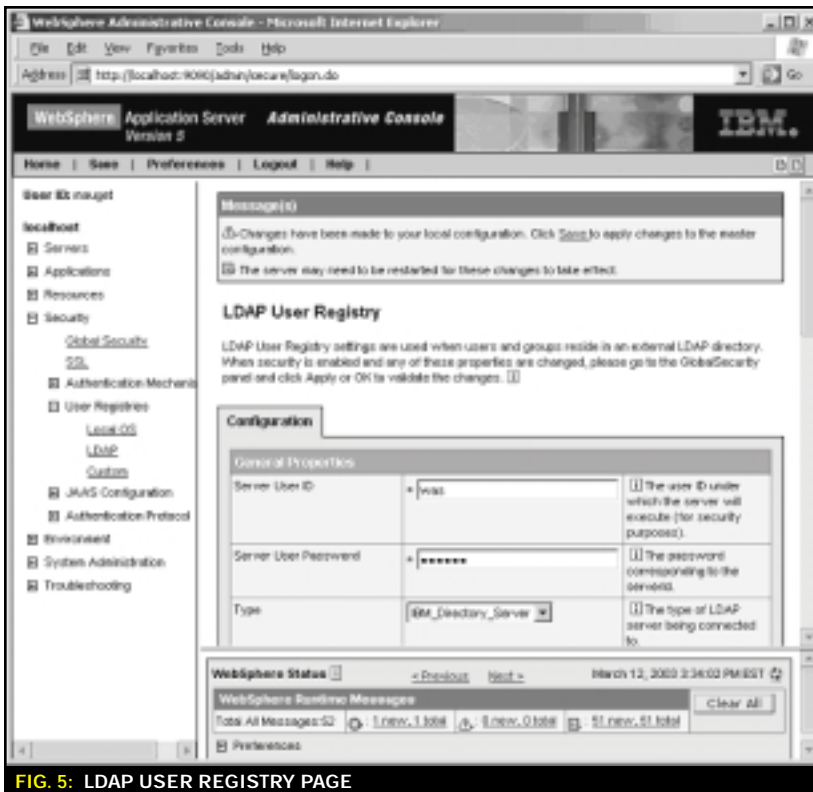


FIG. 5: LDAP USER REGISTRY PAGE

# VERSATA

[WWW.VERSATA.COM/BUSINESSLOGICDESIGNER](http://WWW.VERSATA.COM/BUSINESSLOGICDESIGNER)



FIG. 6: BROWSER LOGIN DIALOG

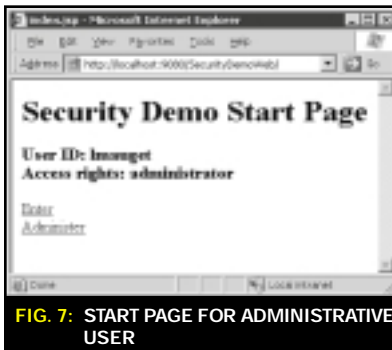


FIG. 7: START PAGE FOR ADMINISTRATIVE USER

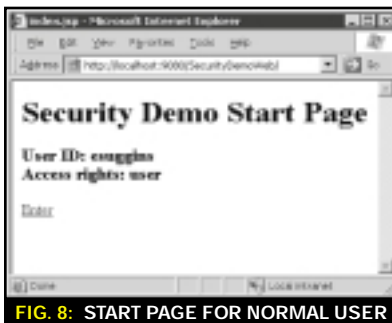


FIG. 8: START PAGE FOR NORMAL USER

6. Press Apply.
7. Press Save in the Message(s) box.
8. Save to Master Configuration.
9. Press Logout and then close the browser.
10. Return to the WebSphere Studio Servers view and restart the server.

The Console view will show security initialized with LTPA and the host and port of your IBM Directory Server as the security principal.

### Run the Application

You are ready to demonstrate LDAP-based authentication and authorization. Follow these steps:

1. In WebSphere Studio, open the Servers Perspective.
2. In the Servers view, right-click WebSphere V5.0 Test Environment, and choose Publish.
3. Right-click the server again and choose Restart. Wait for the console message: Server server1 open for e-business
4. Open an external browser to <http://localhost:9080/SecurityDemoWeb>. The browser security dialog appears because basic authentication was configured for the Web application.
5. Enter "lmaugest" and "password" for the user ID and password (see Figure 6).

6. Press OK. The browser will render the start page (see Figure 7).
7. Notice that you have administrator access because of the LDAP group membership of the user ID lmaugest. Click the Administer link to satisfy yourself that you have access to that page.
8. Close the browser. This destroys the session logon token, effectively logging you out.
9. Again start a browser on <http://localhost:9080/SecurityDemoWeb>. This time log in as "esuggins" with a password value of "password".
10. This time you receive normal access rights and do not see a link to the administration page (see Figure 8).

The administration link is not visible, but can you access its page anyway? Try it. Enter <http://localhost:9080/SecurityDemoWeb/admin.jsp> as the browser address to try to access the page directly. You receive an access violation, as shown in Figure 9. Your production applications can prevent your users from seeing this ugly screen if you configure the deployment descriptor to show a more friendly error page for HTTP status 401.

In addition, the WebSphere Studio Console view shows the following message:



FIG. 9: INSUFFICIENT ACCESS RIGHTS

```
SECJ0129E: Authorization
failed for esuggins while
invoking GET on
default_host:/SecurityDemoWe
b/admin.jsp, Authorization
failed, Not granted any of
the required roles: admin
```

This is helpful in debugging authorization problems because it shows the role that was not granted to the given user ID.

### Configure Secure Sockets Layer

Congratulations! You have configured WebSphere to authenticate users via your LDAP user registry and authorize them to roles defined by LDAP mappings. There is still a data integrity and confidentiality exposure on the network.

# KENETIKS

[WWW.KENETIKS.COM](http://WWW.KENETIKS.COM)

Passwords are happily flowing in clear text from the browser to WebSphere and from WebSphere to the LDAP directory. The cure is to enable Secure Sockets Layer (SSL) on both transports.

#### BROWSER-TO-APPLICATION SSL TRANSPORT

Impose an SSL connection between the browser and any resource associated with the roles user and admin. Simply open the SecurityDemoWeb deployment descriptor to the Security tab. Then set the User Data Constraint to Confidential for the two constraints associated with those roles. Restart the server and access <http://localhost:9080/SecurityDemoWeb> from a fresh instance of your external browser.

The SSL certificate exchange causes a browser security alert because the server's dummy certificate name, jserver, probably does not match your site name (see Figure 10). For production, you would stop this alert by installing a certificate named for your site and signed by a trusted certification authority known to the browser.

For this demonstration, press Yes to accept the server certificate and proceed to the login screen. Log in as one of your defined users.



FIG. 10: SECURITY ALERT

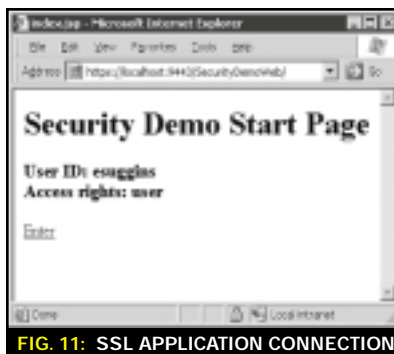


FIG. 11: SSL APPLICATION CONNECTION

Notice that the URL protocol in the browser address field changes to an https protocol and the port becomes the https port 9443 defined in the test server. In addition, a lock icon appears on the status bar of the browser (see Figure 11).

#### WEBSPPHERE-TO-LDAP SSL TRANSPORT

Now passwords and other data flowing between the browser and WebSphere are safe from snooping and undetected alteration, but protection during the transport between WebSphere and LDAP is lacking. Another SSL channel is needed.

This task mainly consists of giving WebSphere a certificate so it can identify and trust the LDAP server. The unattended LDAP channel cannot present security alerts for manual intervention as the browser does. Configure a key file for your LDAP server, create a certificate in it, and then export that certificate to the WebSphere trusted certificate database.

WebSphere Studio can display detailed instructions to follow for this procedure. Display them using the following steps:

1. Click the main menu Help item.
2. Search for: "Configuring SSL for LDAP".
3. Choose the result near the top, labeled "Configuring SSL for the LDAP client: WebSphere Application Server".

Substitute real host names where relevant. The following high-level steps summarize the procedure:

1. Generate a key file for use by your LDAP server.
2. Generate a self-signed certificate within that key file (or request one from a certificate authority).
3. Export the certificate to a flat file.
4. Import the certificate flat file to the trust file on WebSphere.

After configuring the certificates, use the following steps to enable SSL in the server:

1. Ensure that WebSphere is running.
2. Invoke <http://localhost:9080/admin> from a browser to open the WebSphere Administration Console. Log in as was/password, as defined in your LDAP server.

3. Expand the tree to node Security|User Registries|LDAP.
4. Alter field Port to 636.
5. Set option SSL Enabled.
6. Log out.
7. In WebSphere Studio, use the Server view to restart the server.

Use a browser to access the application. The application should operate as before, except that the back-end transport channel between WebSphere and the LDAP server is secure.

### Conclusion

This two-part series demonstrated how to create a WebSphere application that is secured through LDAP and SSL. J2EE applications can assign roles to constraints used to secure resources by the URL or HTTP method. A secured J2EE server, such as WebSphere, maps those roles to principals in some registry. WebSphere LTPA can be configured to use an LDAP server as its principal registry. Advantages include centralized user management, the capability to use the LDAP directory for centralized organizational data, and the capability to enable single sign-on for applications across a set of servers. The directory server controls selective access to LDAP entries such as passwords. WebSphere, the directory server, and the client browser can all use SSL transport to secure data on the transport layer.

### Resources

- Nilsson, D., and Mauget, L. (2003). *Building J2EE Applications with WebSphere*. John Wiley & Sons.
- High, R., Rochat, K., and Knutson, J. (2003). *Professional IBM WebSphere 5.0 Application Server*. Wrox Press Inc.
- Barlen, T., et al. (2002). *Implementation and Practical Use of LDAP on the IBM iSeries Server*. IBM Redbooks.
- Howes, T, et al. (1998). *Understanding and Deploying LDAP Directory Services*. New Riders Publishing.
- Howes, T, and Smith, M. (1997). *LDAP: Programming Directory-Enabled Applications With Lightweight Directory Access Protocol*. Que.

# QUEST SOFTWARE

[HTTP://JAVA.QUEST.COM/QCJ/WDJ](http://java.quest.com/qcj/wdj)



## Part 2: EJBs, JDBC, and application development and monitoring

# WebSphere and Database Performance Tuning

BY MICHAEL S. PALLOS

In a business environment defined by requirements to elevate service levels while reducing IT operational costs, developers seek proven strategies to optimize the production runtime environment of their WebSphere Application Server implementations. The nearly universal objective of IT leaders today is to improve application performance and maximize the environment's ability to support broad business objectives.

In the May issue of *WebSphere Developer's Journal* [Vol. 2, issue 5], I presented the first of a two-part series that provides strategies for tuning network and database interfaces to optimize IBM WebSphere Application Server implementations. That article discussed best practices for database connection pooling, prepared statement cache, and session persistence. In this article, I present best practices and tuning techniques for EJBs (Enterprise JavaBeans), JDBC (Java Database Connectivity), and application development and monitoring.

### Enterprise JavaBeans

The EJB specification is the foundation for Java 2 Enterprise Edition (J2EE), offering component services such as distributed transactions,

security, and life-cycle management. There are two types of EJBs – session beans and entity beans (J2EE 1.3 also includes message-driven beans). A session bean instance belongs to a specific user and maintains the user's state as he or she interacts with the Web site. User status information, however, is lost in the event of a catastrophic system failure. A shopping cart utilized by an online shopper exemplifies the use of a session bean. As the user progresses through the Web site, he or she can add and remove items from the cart – the session bean. Once the user makes a purchase or chooses to leave the Web site without completing a purchase, the user's interaction with the site is terminated and the session bean is destroyed.

An entity bean represents data – typically a row in a database – and

can be shared among multiple users. Continuing with the shopping cart example, if the user makes a purchase, the data will persist to a database via an entity bean. Should the transaction require the creation of an invoice, an entity bean would be created containing the user's data. Because it is persisted to the database, entity bean data will survive a system crash.

EJBs offer many advantages, including the ability to leverage distributed objects and transactional services. These advantages, however, come at a price – increased complexity. EJBs that are architected or incorporated incorrectly can have an adverse impact on application processing. Proper EJB utilization and optimization, conversely, reduces processing cycles, which, in turn increases application performance and enhances the end-user experience.

EJB best practices that optimize the storage and retrieval of persistence data include the following:

- **Use the appropriate isolation level.** Isolation levels are used to restrict access to a resource to which other concurrent transactions have access. They allow users to lock down or isolate shared database resources to four levels of granularity:
  - *Read Uncommitted:* The transaction can read uncommitted data from other transactions. Uncommitted transactions yield the lowest overhead since the container is doing the least amount of work. Imagine, for instance, that a user reads the database and retrieves certain values. Another user then starts a BEGIN WORK transaction and inserts values into the database. If the first user performs another database read before the second user has executed a COMMIT WORK or ROLLBACK command, he or she will read the uncommit-



ted data that has been temporarily inserted into the database. The scenario just described is also known as a "dirty read."

**–Read Committed:** The transaction can read committed data only. If we repeat the scenario described above with a transaction state set to Read Committed, a user performing a read on a database containing data from a BEGIN WORK that has not been COMMITTED will not be able to access the uncommitted data. The user can only read committed data. A Read Committed is more restrictive than a Read Uncommitted isolation level, requiring the system to work harder. A Read Committed status, therefore, has a greater impact on processing than a Read Uncommitted status.

**–Repeatable Read:** The transaction is guaranteed to read back the same data on each successive read. A Repeatable Read is more restrictive than a Read Committed, having an even greater impact on processing.

**–Serializable:** All transactions are serialized or completely isolated from each other. A Serializable transaction is the most restrictive isolation level, and has the greatest impact on processing. An EJB that uses a serialized transaction isolation level is guaranteed to achieve consistent results from the database since the database is locked from other users, essentially creating single-threaded processing. While rookie developers are often tempted to incorporate the serializable isolation level to ensure data integrity, it has major constraints. Although results are guaranteed, all concurrent users are locked out of the database, which slows down the application. To optimize performance for data access, developers must fully understand the difference between the four isolation levels and their indicated uses.

- **Carefully define access intent.** The access intent attribute contains one of two states: (1) read and (2)

update, with the default setting being update. For methods that are going to be READ ONLY, setting access intent to READ will increase data access performance. Under such conditions, the database is accessed with an intent to read – and not update – thus removing unnecessary database locking.

## Java Database Connectivity

JDBC provides a standard library for accessing relational data, allowing users to develop SQL calls using the Java Application Programming Interface (API). The JDBC driver is responsible for the data access communication interface with the database management system (DBMS).

Selecting the appropriate driver is fundamental to improving performance. According to Sun Microsystems, there are four categories of JDBC drivers (Types 1–4), and more than 177 models.

- **Type 1 – JDBC–Open Database Connectivity (ODBC) bridge:** A Type 1 driver provides JDBC access to one or more ODBC drivers. Type 1 drivers assist companies that already possess a large ODBC population with the JDBC educational process. Type 1 drivers are slow, however, because they require JDBC-ODBC translation. As such, they are not well suited for large enterprises.
- **Type 2 – Partial Java driver:** A Type 2 driver converts the calls made from the JDBC API to the receiving machine's API for a specific database (DB2, Oracle, Sybase, SQL Server, etc.). A Type 2 driver contains compiled code for the back-end system. Type 2 drivers process more quickly than Type 1 drivers. The code must be compiled, however, for every operating system on which the application runs. In Windows NT and z/OS development, organizations may wish to develop with a Type 4 driver and then transition to a Type 2 driver for production.
- **Type 3 – Pure Java driver for database middleware:** A Type 3 driver provides connectivity to

many different databases, translating JDBC calls into the middleware vendor's protocol and then into the database-specific protocol via the middleware server. A Type 3 driver is often faster than Type 1 and 2 drivers, and is useful if an organization wishes to connect to multiple database types. Database-specific code, however, must reside on the middle tier. If the application is going to run on different operating systems, a Type 4 driver may be more appropriate.

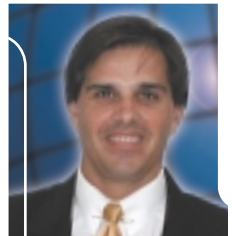
- **Type 4 – Direct-to-database Java driver:** A Type 4 driver converts JDBC calls into packets that are transferred over the network in the database's proprietary format, allowing a direct call from the client to the database without a middle tier. Type 4 drivers often offer better performance than Type 1 or 2 drivers; do not require additional code on client or server machines; and can be downloaded dynamically. Type 4 drivers, however, are not optimized for the operating system and are unable to take advantage of operating system features. Users also need a different driver for each different database.

When developing on multiple platforms, many aspects of testing can be simplified by using a Type 4 driver. A Type 2 driver can be incorporated after system testing and production is completed. According to Sun Microsystems and IBM, Type 2 JDBC drivers offer the best performance when incorporating distributed transactions and catalog databases.

When selecting a driver, developers should also consider the JDBC API levels that it supports. Currently, there are two JDBC API levels. While JDBC 1.0 is the default, WebSphere requires JDBC 2.0. The JDBC API best practice is to incorporate version 2.0.

## Application Development and Monitoring

WebSphere Application Server development and monitoring tools



### ABOUT THE AUTHOR

Michael S. Pallos, MBA, is a senior solution architect for Candle Corp. ([www.candle.com](http://www.candle.com)) and has 18 years of experience in the IT industry. He is a consultant to some of Candle's largest corporate customers, a featured speaker at industry conferences, and a doctoral student at Walden University. Candle, based in El Segundo, California, provides solutions to manage and optimize business-critical infrastructures, systems, service levels, and other information technology assets.

### E-MAIL

[michael\\_pallos@candle.com](mailto:michael_pallos@candle.com)

can offer insight into potential application bottleneck areas that require additional attention to achieve optimal performance. An ideal development tool identifies workflow analysis down to the method and thread level, offering real-time and historical information. Monitoring tools should also be able to report results on all elements contained within the application. These elements include the end user's perspective through WebSphere Application Server, DBMS, transport layers, connection layer, and, if incorporated, any legacy system components.

Three best practices to consider when selecting and/or deploying monitoring capabilities follow:

- **Select a development tool that delivers granular, real-time information.** During development, a tool can help to identify and/or eliminate memory leaks, bottlenecks, and optimization. Development tools should offer granularity to the method level and thread level, SQL calls, and heap insight. The tool should also provide the ability to look into the application during execution and deliver sufficient analysis to allow developers to proactively address problems.
- **Select monitoring tools that assist with the early stages of the development phase.** Monitoring tools should be capable of capturing the complete end-to-end scenario, including the end-user experience, latency time, and the performance and availability of

systems external to WebSphere Application Server, such as CICS, DB2, WebSphere MQ, and B2B. Pure Java monitoring tools are excellent for initial development. However, many WebSphere Application Server applications incorporate external Java components to complete application processing. A desirable monitoring tool not only reports on the Java pieces of the application, but the entire application portfolio. If a highly productive Java application is executing on a machine that is underutilized, the operating system monitoring agent can then report the problem to the Java developer or operations personnel.


- **Select a solution that offers both development and monitoring tools.** (One choice could be the Candle PathWAI solution suite.) This strategy simplifies training, development, and production rollout.

### Conclusion

Today's developers face unprecedented pressure to create WebSphere applications that roll out rapidly and flawlessly, and adhere to business service-level requirements for processing and end-user experience. Once deployed, the challenges begin anew, as developers, system administrators, and operations work to optimize the production runtime environment for their WebSphere Application Server implementations. Best practices for tuning

WebSphere Application Server persistence layers, performance, and database interfaces represent the most direct and hazard-free path to optimizing WebSphere Application Server implementations.

### References

- Alur, N., Lau, A., Lindquest, S., and Varghese, M. (2002). "WebSphere Application Server and DB2 UDB Performance." *DB2 UDB/ WebSphere Performance Tuning Guide*. IBM Redbooks.
- Endrei, M., Cluning, R., Daomanee, W., Heyward, J., Iyengar, A., Mauny, I., et al. (2002). *IBM WebSphere V4.0 Advanced Edition Handbook*. IBM.
- Erickson, D., Lauzon, S., and Modjeski, M. (2001, August). *WebSphere Connection Pooling*: [www-3.ibm.com/software/web/servers/appserv/whitepapers/connection\\_pool.pdf](http://www-3.ibm.com/software/web/servers/appserv/whitepapers/connection_pool.pdf)
- Hutchison, G. (2002). *DB2/WebSphere Integration*: [www.websphere-users.ca/presentations/hutchison.PDF](http://www.websphere-users.ca/presentations/hutchison.PDF)
- Monson-Haefel, R. (2000). *Enterprise JavaBeans*. O'Reilly & Associates.
- Silberschatz, A., Korth, H., and Sudarshan, S. (2002). *Database System Concepts*. McGraw-Hill Higher Education.
- *Java 2 Platform, Standard Edition, v1.3.1 API Specifications*: <http://java.sun.com/j2se/1.3/docs/api>
- *JDBC Data Access API, JDBC 2.0 Optional Package API*: <http://java.sun.com/products/jdbc/index.html>
- *JDBC Data Access API Drivers*: <http://industry.java.sun.com/products/jdbc/drivers> 

**"WebSphere Application Server development and monitoring tools can offer insight into potential application bottleneck areas that require additional attention to achieve optimal performance"**

# DIRIG SOFTWARE

[WWW.DIRIG.COM](http://WWW.DIRIG.COM)

# No Portal Is an Island

Integrate existing applications into a portal framework

— BY ZHE WANG AND QI LI —

A portal delivers enormous value to IT organizations, enabling centralized access to applications and personalized information in order to gain a simplified infrastructure, faster development, and enhanced employee productivity. Companies can leverage and maximize their portal framework by integrating existing applications; aggregating data, content, and processes from them; and providing users with dynamic, collaborative, and personalized views into the applications.

Integrating existing applications can be as simple as running the legacy application and content within the portal frame or as complicated as wrapping the existing Web application into a portlet. This article will describe the common methods available for integrating applications into a Portal Server environment, how they are implemented, and the pro and cons of each option. The sharing of data between the integrated applications will be saved for a future article.

## Overview of Integration Strategies

We will present six integration methods in order of increasing complexity and control over the depth of integration (see Figure 1). As we detail each method we will highlight whether the portal developer requires access to modify the target application, how each method may conform to the organization's time-to-market constraints, and the depth of integration offered, both technically and visually.

### Method 1: The Out-of-the-Box Method

As more and more organizations adopt portal-based architectures for their intranet, Internet, and extranet presences, vendors are readily making their solutions available as prebuilt portlets. These prebuilt portlets can provide automatic integration with sophisticated back-end systems such as ERP, CRM applications, and finance and HR systems. In addition, WebSphere Portal Server comes with some fairly complete solutions for integrating with Microsoft desktop environments, including Exchange, Office, and Document viewers. In general, to use these portlets you simply drop them in, configure, and you're done.

## ABOUT THE AUTHOR

Zhe Wang is a member of the Prolifics Technical Services team, a team of experts dedicated to helping customers design, implement, and deliver distributed transactional applications. She is responsible for assisting customers with their J2EE and WebSphere Application Server requirements, and specializes in portlet design and building WebSphere application components.

## E-MAIL

zwang@prolifics.com

The entrée into portal development typically begins with building portlets and displaying them within a portal interface. However, this article focuses on a very important part of portal implementations – the reuse of legacy applications – and provides key integration strategies. There are several common methods for integrating applications into WebSphere Portal Server. No single method is better than the others, and your specific integration circumstances will dictate which is most appropriate.

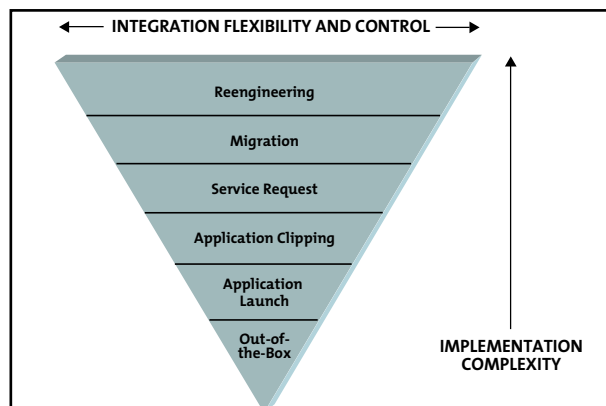


FIG. 1: INTEGRATION METHODS

### Out-of-the-Box

Legacy applications available as prebuilt portlets to be dropped into portal

- ✓ Complete, robust solution
- ✓ No coding necessary
- ✓ Simple to implement
- ✓ Quick to market



There is very little control over the inputs and outputs of the business services. However, with pre-built portlets you gain a complete solution with no coding necessary – a very simple, fast path toward deployment.

WebSphere Portal Server comes with some prebuilt portlets. Additionally, the IBM Portlet Catalog, [www-3.ibm.com/software/webervers/portal/portlet/catalog](http://www-3.ibm.com/software/webervers/portal/portlet/catalog), contains several valuable portlets for free or fee that are easy to download.

## Method 2: The Application Launch Method

When there are no prebuilt components available for an application desired for integration, it can still be integrated by simply launching an instance from the portal to give the end user access to the application's functionality. The entire application can be launched to sit directly on a piece of real estate within the portal page using an IFRAME (inline frame). Multiple applications can be directed to multiple frames on the same page. Alternatively, although it may give a less integrated feel, the application can be launched outside of the portal framework as a free-floating window.

### Application Launch

Full application runs visually within a portlet or launched outside portal window

- ✓ Simple to implement
- ✓ Target application remains intact, functioning as before
- ✓ Application appears within unified portal view

The launched application operates independently of the rest of the portal, allowing it to be integrated without modification. This is extremely important since it ensures that navigational links within the integrated application will not cause the user to be navigated away from the portal.

To launch an application, we recommend using the WebPage portlet that comes with WebSphere Portal Server and is designed to display content from any Internet or intranet page. The WebPage portlet is simple to use and offers flexibility. It can handle several points of access into the application, including display of an HTML page, execution of a JSP and display of its output, or invocation of a servlet. However, WebSphere Portal does include the JSPServer portlet and ServletInvoker portlet, which are quick and convenient for direct invocation of JSPs and servlets.

Steps to follow when using the WebPage portlet include:

1. Create an instance of the WebPage portlet specific to the target application.
2. Edit the parameters to set key values that define the style, URL, and URI to launch, such as useIFrame, defaultURL, viewURI, helpURI, and flatViewURI (see Figure 2)
3. Control the look and feel of the IFRAME by setting parameters, including width and height.

Note that because portlets have restrictions on generating some HTML tags (e.g., body and title tags) you may need to modify existing Web resources if their responses contain these tags. For performance reasons it is not advisable to parse and remove these tags from the portlet at runtime. Other potential limitations include:

- Browser support for IFRAME first appeared in Internet Explorer 5.1 and Netscape 6.0.
- IFRAMEs containing applets will not work correctly in a portal. For every request the Portal Server sends back to a new portal page, the old applet instance stops and a new one starts.
- Site navigations and state of actions are not stored. A browser refresh will take you back to the initial screen, since the page associated with the IFRAME URL is simply reloaded.
- No support for alternate markups (e.g., WML).

In summary, launching an application into a WebPage portlet is the simplest approach when you do not have access to the target application for modification. Visually, the application is integrated into the unified view offered by the portal framework, although the application's look and feel itself may not match that of the rest of the portal. Control is turned over to the application itself, away from the Portal Server domain, enabling the application to function normally.

Portlet Name: Meeting Room Scheduler -- WebPagePortlet

Edit Parameters:

Parameter	Value	✖ Delete
<input checked="" type="checkbox"/> useIFrame	Yes	
<input type="checkbox"/> defaultURL	http://host:port/...	
<input type="checkbox"/> width	100%	
<input type="checkbox"/> viewURI	/WEB-INF/html/jsp/meeting_room/...	
<input type="checkbox"/> scrolling	Auto	
<input type="checkbox"/> helpURI	/WEB-INF/html/jsp/meeting_room/...	
<input type="checkbox"/> editScreenClassName	com.ibm.wps.portal.frame.Wps...	
<input type="checkbox"/> flatViewURI	g/WpsWebPageEditFrameView.jsp	
<input type="checkbox"/> height	200	
<input type="checkbox"/> editURI	/WEB-INF/html/jsp/wpsWebPage/...	

✚ Add

Edit Locale Specific Titles:

Locale	Title	☑ Set title for selected locale
<input type="radio"/> Simplified Chinese	Web 页面 Portlet	
<input checked="" type="radio"/> English	Meeting Room Scheduler	

FIG. 2: WEBPAGE PORTLET CONFIGURATION



### ABOUT THE AUTHOR

Qi Li is a senior consultant at Prolifics working in the WebSphere Consulting division, a staff of specialized consultants retained by IBM to solve the toughest of their customer's challenges and deliver training, mentoring, and development services. An expert with WebSphere Portal software, Qi has helped organizations such as CIBER, GE, CommercialWare, UPS, MetLife, and Salomon Smith Barney with their WebSphere and Portal initiatives.

### E-MAIL

qli@prolifics.com

### Method 3: The Application Clipping Method

While the previous method launches entire applications into the portal, in some cases it is desirable to integrate only a portion of an existing Web application. WebSphere Portal Server offers an easy, intuitive way to clip content and functionality from existing Web sites and present them in custom portlets. With application clipping, also known as Web clipping, the user has the freedom to keep the important components on the page while discarding undesirable information or elements from the document that the client device is incapable of displaying. This simplifies what is being sent to the device while reducing the amount of data being transmitted, a particularly useful aspect when wireless devices are involved.

#### Application Clipping

All or a portion of an external application is 'clipped' for display in a portlet

- ✓ Simple to implement
- ✓ Target application remains intact, functioning as before
- ✓ Able to select desired portions of the application to display

For this method, WebSphere Portal Server provides the Web Clipping Portlet. Web clippers can also be configured to access content through a firewall using a proxy, such as HTTP or SOCKS, and link-traversal behavior can be controlled with URL rewriting. The Web clipping process will generate a "Web Clipper", which is a concrete portlet of the Web Clipper runtime portlet parameterized by an XML configuration stored in `com.ibm.wps.portlets.clipper.config.data`.

To use the Web Clipping Portlet, set the following configurations (illustrated in Figure 3):

1. Modify the Clipping Type to select from one of the two following options:
  - **HTML Clipping:** HTML Clipping operates on the DOM form of the source document by using node delimiters. This choice enables you to manually select elements of the document by clicking on them with the mouse.
  - **Text Clipping:** Text Clipping operates on the text form of the document by using string-based delimiters. This choice enables you to select the content between specific text strings in the HTML document. Content between these strings is kept, and all other content is discarded.



FIG. 3: WEB CLIPPING PORTLET CONFIGURATION

2. Modify Firewall Options if the content is within a firewall.
3. Modify Authentication Options when the content requires security verification for access.
4. Modify Rules for URL rewriting to change the way URLs are handled by the Web clipper, modifying the URL from the content source. The URL rewriter can be controlled by two sets of regular expressions called rules. The URL rewriting component is called after the Web content has been clipped, but before any further transcoding occurs. It attempts to match each URL in the document against a rule in one of the sets.
5. Modify Security Options when you want to remove JavaScript from clipped content.

Web clipping is not well suited to dynamic content, and there are certain elements within Web content that cannot be clipped using HTML clipping:

- HTML Clipping for pages containing `<FRAME>` elements is not supported and will generate an error.
- JavaScript may not be clippable. Most JavaScript is evaluated and executed on the client after the clipping process is complete. This means that any content dynamically generated by such scripts cannot be subject to clipping. In general, it is a good idea not to include any JavaScript within a Web clipper unless that JavaScript is self-contained, meaning it does not depend on other variables, methods, structures, or other elements that are defined elsewhere.

The Web Clipping Portlet does allow an enterprise to deploy desired portions of existing content in portlets within WebSphere Portal without having to create new versions of the existing content.

### Method 4: The Service Request Method

Another form of integration is through access to the business services that make up an application, whether they are front-end elements, such as JSPs, or back-end services. Programmers can get limited control over the service return content so that the information can be altered or supplemented before being displayed within the portlet. Additionally, this method is appropriate when a proxy must be used for dynamic delivery of content from an external Web site – for example, when integrating newsfeeds.

#### Service Request

An application is called by a portlet, which controls input/output

- ✓ Simple to implement
- ✓ Target application remains intact, functioning as before
- ✓ Can access output of a remote service for small modifications

WebSphere Portal provides the `ContentAccessService` service that can retrieve content from resources outside your intranet. It can transparently use a proxy server for both HTTP and HTTPS protocols if a connection is needed from a portal to a remote Web site. When creating a portlet to access Web resources, use the following methods of the `ContentAccessService` class:

- The `getInputStream()` method connects to the Web resource specified in the `urlString` and returns the output of the Web resource as a stream object.
- The `getMarkup()` method functions similarly to `getInputStream`, but stores the retrieved stream from a Web resource as a String object.
- The `include()` method uses an object of type `RequestDispatcher` if the resource is internal; otherwise, it uses a URL connection to access an external Web resource.
- The `getURL()` method, which returns a URL object after following redirects, uses a proxy if necessary.

As a limitation, when accessing Web resources the content is simply taken and written to the portlet's output. Therefore all relative links to other pages or images will be broken. This can be solved by parsing the content or using some enhanced browser portlet.

In general, the `ContentAccessService` service is beneficial when you need access to the output of a remote service through a proxy or in order to modify the output before it is written to the page, delivering it dynamically.

### Method 5: The Migration Method

More sophisticated application integration strategies become available if the portal developers have the necessary access to modify the source code for the target application. The simplest approach is to migrate only the application functionality that you require to run within portlets that will be installed into the portal. This is obviously more involved and time-consuming, however, it offers more depth of integration both on the inputs and outputs, as well as the look and feel.

Migration
Desired application code and functionality is ported to portlets
<ul style="list-style-type: none"> <li>✓ Higher degree of control over application input/output</li> <li>✓ Flexibility in look &amp; feel for a more native portal integration</li> <li>✓ Do only portions as needed and as time permits</li> </ul>

During a migration, you encapsulate parts of your Web application within a portlet (wrappering) and modify as required. Portlets are created from servlets; and JSPs are modified so they can be invoked from portlets. The following steps should be performed when migrating legacy applications to run within a portal application:

- **Replace your servlets with portlets:** To portalize an existing Web application, you need to transfer the business logic in the servlet to the portlet. The portlet class can extend the `AbstractPortlet` class and overwrite the `doView()`, `doEdit()`, and `doHelp()` methods. The servlet logic will mainly be implemented in the `doView()` method.
- **Implement your ActionListener class:** There are several characteristics that distinguish servlets from portlets. The presentation layer is one of them. Servlets tend to have an input JSP file that makes a servlet call, and an output file that can invoke other JSP files to drill down to other tasks. Portlets have the same notion but are implemented in a different way. When a portlet is initial-

ly constructed on the portal page for a user, it is displayed in its view mode. This is accomplished by calling the portlet's `doView()` method, which might include a different JSP. All of the JSP files that need to interface with users will be included in `doView()` by checking the data stored in their respective `PortletSession` or objects. There will be an if-else statement to distinguish which JSP should be included to render the markup.

Another characteristic that distinguishes servlets from portlets is the way they process user actions. For example, when a user clicks a link or button in a JSP, this may invoke an HTTP request or call other JSP files. However, when a user clicks a link or button in a portlet, this generates an action event. To receive notification of the event, the portlet must have an event listener registered in the portlet deployment descriptor. The event listener needs to implement the `PortletAction` class that is linked to the specific URL and passed by the `ActionEvent`.

To enable the action event, a return URI is created by using the `PortletURI` class and its `addAction()` methods. This URI is then passed to the JSP as an object contained in the `PortletRequest` object. The `ActionListener` can then take the appropriate action to process the request, package the result in the `PortletRequest` object, and send it back to the calling portlet.

- **Modify your JSP files:** Portlets can only create markup; the Portal framework is responsible for assembling them into a complete page. In addition, portlets have notions of action events, message events, and window events that require the `ActionListener` to perform various tasks. External links in JSP files can remain the same. Once invoked, control is passed to an external site, and the portlet's session may expire at that moment. If you have an internal JSP link that should be included in the `doView()` method and you need to specify the portlet URL that associates the action event with the appropriate `ActionListener` class, use the following code:

```
<ahref="EmployeeDetail.jsp?key=
<%=emp.getEmpno()%>"></a>
And this is in the portlet:
<a
href="<%=empList.getEmpDetailURI()%>?key=<%=
emp.getEmpno()%>"></a>
```

*-JSP parameter passing:* The chaining of JSP files within a portlet is done by calling a specific URI that associates a specific action event with its corresponding `ActionListener`.

*-Button action:* Since a button is processed as an event action, you must specify the portlet URL.

*-Image reference:* To access the images that are stored under the `/images` directory, you must tailor your JSP to use the JSP tag library.

*-Modify form action from the servlet class name to the portlet URI:* To submit a form in the portlet, you must specify the URI that will handle the form input data.

```
Servlet:<form name="search"
action="DirectorySearch" method="get ">
Portlet:<form method="post " name=="form "
action="<%=phoneBook.getSaveURI()%>">
```



### Method 6: The Reengineering Method

Beyond a partial migration of individual business services to portlets is the extreme method of completely reengineering your legacy application to be re-architected for the portal. This offers the ultimate benefit of integrating your legacy applications to be completely native to the portal infrastructure. You are able to reformat the application's user interface to be compliant with the portal interface, reorganize the business logic processing into loosely coupled portlets, and alter the workflow of the application to take advantage of portal navigation and interoperability conventions.

Reengineering
The entire application is ported, reformatted, and restructured to be native to and suitable for a portal infrastructure
<ul style="list-style-type: none"> <li>✓ Total control over application input/output</li> <li>✓ Flexibility in look &amp; feel for a more native portal integration</li> </ul>

### A Real-World Example Application

No one method is better than the rest – in any situation one may be better suited based on unique business requirements, technical requirements, and technical challenges. To demonstrate, let's walk through a scenario.

Our medical services provider, MedCompany, has deployed two portals: a Dynamic Workplace B2E portal for use by employees, and the MedXchange B2B portal for use by physicians. In MedXchange they desired a capability to provide a search within a medical library provided online by Medical Research Labs. Additionally they desired access to an advanced search capability from within the medical library view, provided as a link to the existing “Med Explorer” library application.

So here we have two Web applications that already exist that we would like to leverage in our portal. We need to modify the output from the medical library only slightly to add the advanced search capability, but we have no developer access to the target application. Given this criteria, it makes sense to integrate the applications using the Service Request method, which enables us to leverage an existing Web application service very quickly and simply, while modifying the output slightly.

To do so, the ContentAccessService service was customized to let patients or physicians search the Medical

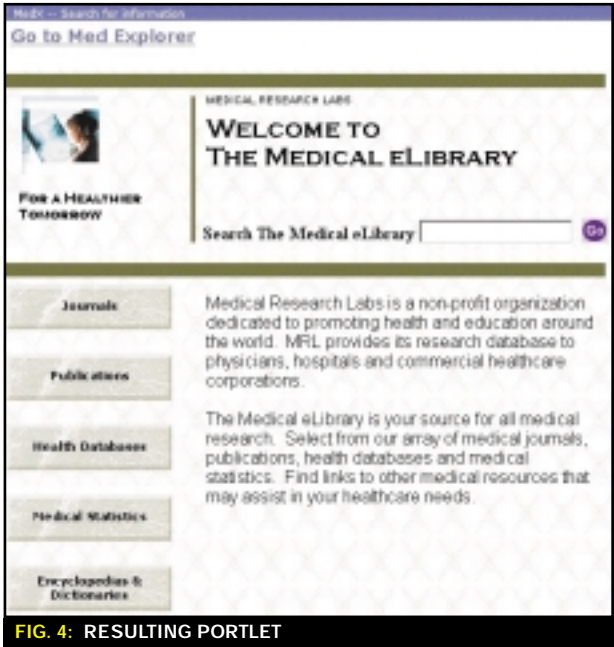



FIG. 4: RESULTING PORTLET

Research Labs eLibrary page, adding in an extra link to provide access to Med Explorer for the more extensive advanced search. Listing 1 details the code that uses ContentAccessService for this purpose and Figure 4 shows the resulting portlet.

### Conclusion

As in most application development efforts, integration of legacy assets is a key component of developing and deploying effective portal solutions. Understanding the current IT environment and the business and technical requirements for the portal delivery will determine which of many integration techniques is best for each integrated application. It is important to consider the overall goals of the portal, such as the desired level of appearance of unified view, and time-to-market constraints. It is also important to understand what access the portal developers have to modifying any of the target applications, and what input/output manipulation is desired as well. Once applications are linked, however, you will want them to start sharing application data such as authentication information, so that users can leverage single sign-on to the portal for all the integrated applications. But alas, we must save that topic for a future article. 

<p>LISTING 1: CODE USING CONTENTACCESSSERVICE SERVICE</p> <pre> public class MyContentAccessServicePortlet extends PortletAdapter {     public void doView( PortletRequest portletRequest,         PortletResponse portletResponse)         throws PortletException, IOException     {         PortletContext context =             this.getPortletConfig().getContext();         PrintWriter pw = portletResponse.getWriter();         //Reads url as configuration parameter         String url = </pre>	<pre> portletRequest.getPortletSettings().getAttribute("url");          // Accesses ContentAccessService          ContentAccessService service = (ContentAccessService) context.             getService(ContentAccessService.class);         pw.println("&lt;a href=\"http://www.medexplorer.com/\"             target=\"_blank\"&gt;&lt;h3&gt;Go to Med Explorer&lt;/h3&gt;&lt;/a&gt;");         //getMarkup opens a URL connection and returns output of             URL as String         String servletOutput =             service.getMarkup(url,portletRequest,portletResponse);         pw.println("&lt;b&gt;" + servletOutput + "&lt;/b&gt;");     } } </pre>
--	---



# GLOBAL KNOWLEDGE

[WWW.GLOBALKNOWLEDGE.COM](http://WWW.GLOBALKNOWLEDGE.COM)

# CANDLE

[WWW.CANDLE.COM](http://WWW.CANDLE.COM)

# CANDLE

[WWW.CANDLE.COM](http://WWW.CANDLE.COM)

*Excerpted from Chapter 4: Debugging in  
WebSphere Studio Application Developer*

# Building J2EE Applications with IBM WebSphere



## ABOUT THE BOOK

*Building J2EE Applications with IBM WebSphere*, book and CD-ROM by Dale R. Nilsson and Louis E. Mauget  
ISBN: 0471281573  
Publisher: John Wiley Publishing  
List Price: \$50  
Available: June 2003

From *Building J2EE Applications with IBM WebSphere* by Dale R. Nilsson and Louis E. Mauget. Copyright © 2003 by John Wiley Publishing, Inc. All rights reserved. Reproduced here by permission of the publisher.

WebSphere provides a host of features for developing and deploying J2EE applications. With this in-depth book, you'll quickly learn how to combine the power of J2EE with WebSphere, according to the publisher. The book covers all aspects of J2EE development, including J2EE architecture, Java applications, Java servlets, JavaServer Pages, JDBC, Enterprise JavaBeans, XML, XSL, and Web services. The following excerpt from Chapter 4 focuses on the WSAD debugger.

**Y**ou have seen how to add to code to aid in debugging. How would you debug code if you could not modify it? WSAD includes a debugger that enables you to detect and diagnose errors in a Java program that is running either locally or on another computer. You may monitor and control the execution of a multithreaded application by setting breakpoints, where you can step into, through, or over Java methods. You may examine the contents of variables while stopped at a breakpoint or may suspend threads and suspend the launching of subsystems.

## Debug Preparation

The debug target need not be modified, but it is helpful to use a coding style that does not place

multiple statements on one line. Some of the debugger features, such as breakpoints, operate on a line-by-line basis. In addition, it is helpful to attach source code for supporting JAR files when it is available. Then, you could step through that code, or place breakpoints in it, instead of being constrained to skip over it in the debugger.



FIG. 4.1: CALCULATOR APPLICATION

## Import Sample Projects

Let's use a real program that has actual problems for test purposes. Import the buggy-calculator-app and buggy-calculator-lib projects found on the Web site for this book. You shall reference code in these projects as you survey debugging in WSAD. The calculator is a Java Swing application shown in Figure 4.1.

You will find and repair three problems in the calculator. Its overall model-view-controller structure is diagrammed in Figure 4.2. Class Calculator is the view. The controller is the actionPerformed() method, which routes events to application logic written in the mediator design pattern.

The mediator contains the state of the application and dictates its flow. It is diagrammed in Figure 4.3. The mediator is not dependent upon any particular view technology. It is capable of interacting with a Swing class as well as a command-line view, a JavaServer Page, a Java servlet, or a Java applet.

The mediator uses model or domain logic housed in the CalculatorEngine class. See Figure 4.4. Methods of this class return a result in a JavaBean suitable for rendering by any kind of Java view technology.

The calculator diagrams should show a high-level concept of the logic. You shall investigate three problems in the mediator in the following sections.

## Breakpoints

What exactly is breakpoint? It is a temporary marker that you may logically insert into an application to signal to the debugger to stop thread execution at that point. The insertion does not physically modify the application. The debugging engine works in concert with the VM to implement a breakpoint. An executing thread is suspended as it attempts to pass through an

enabled breakpoint. At this point, you may view the stack of nested method calls to reach that point in the thread. Additionally, you may inspect the contents of variables at any level in the stack. You can step over statements, step into or over subsequent methods, resume running until the next breakpoint is reached, or resume running until the logical end of program execution.

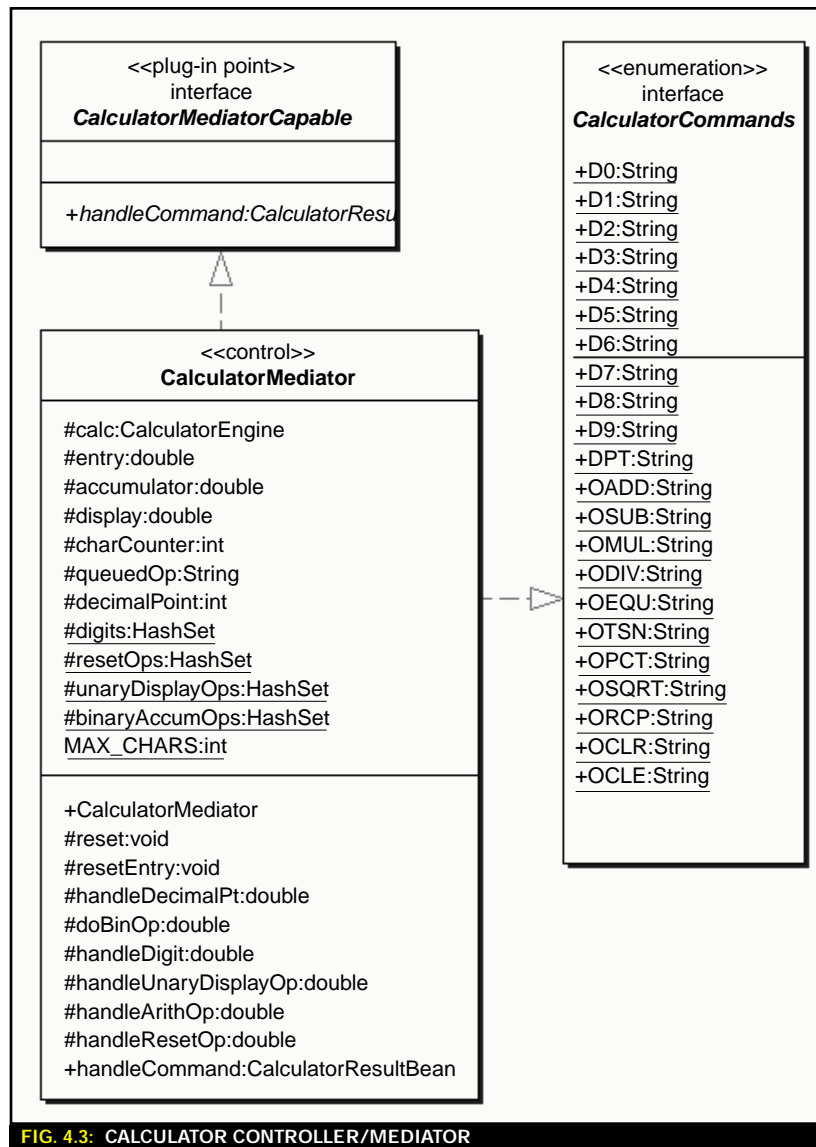
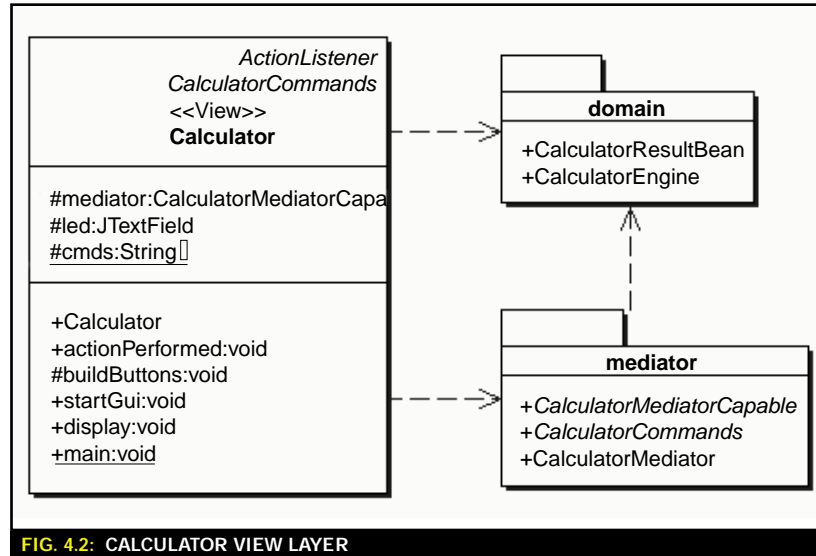
An enabled breakpoint displays as a small blue circle in the editor marker bar or the Breakpoints view. A blue breakpoint icon means that the breakpoint is unverified. This means that its target class has not successfully been loaded into a Java VM. When it is successfully installed in a class in a VM at run time, it becomes a green circle. You may temporarily disable a breakpoint such that it will not suspend a thread attempting to pass through it. A disabled breakpoint icon is rendered as a white circle in the editor marker bar or the Breakpoints view.

There are general line-oriented breakpoints and exception breakpoints. You can set line breakpoints to be armed after a certain hit count is reached. You will learn how to manage these breakpoints.

#### ADD A BREAKPOINT

A breakpoint is set on a single executable line of an object. This line-oriented nature is the reason it is better to have one statement per line, aside from this being a good practice for readability.

Add a breakpoint by opening the file where you want to set the breakpoint. Choose the line where you want execution to stop. The vertical bar to the left of the text is called the marker bar. Right-click the marker bar at the desired line, and then choose Add Breakpoint, as seen in Figure 4.5, or you may simply double-click the marker bar at the desired line to add the breakpoint. Try setting a breakpoint in the main() method of bellmeade.calculator.Calculator in project buggy-calculator-app.



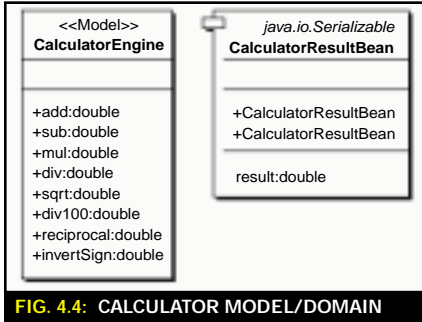


FIG. 4.4: CALCULATOR MODEL/DOMAIN

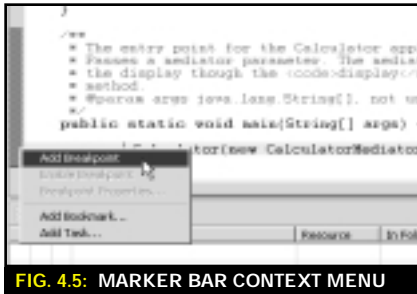


FIG. 4.5: MARKER BAR CONTEXT MENU

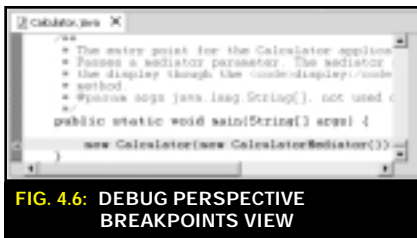


FIG. 4.6: DEBUG PERSPECTIVE BREAKPOINTS VIEW



FIG. 4.7: BREAKPOINT HALT

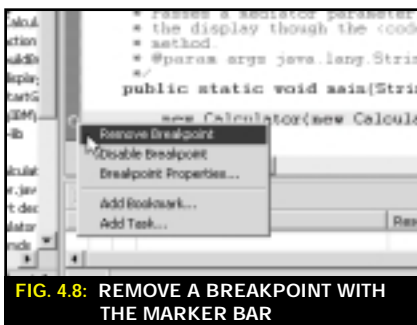


FIG. 4.8: REMOVE A BREAKPOINT WITH THE MARKER BAR

A new breakpoint marker appears on the marker bar, directly to the left of the line where you added the breakpoint. In addition, the new breakpoint appears in the Debug Perspective Breakpoints view, as seen in Figure 4.6.

Run the class as a Java application. Why did it not stop at the breakpoint? You have to run it in a separate debugging VM. Use the debug icon to the left of the run icon. Select Debug as a Java application. This time thread execution should be suspended before the enabled breakpoint line of code is executed. The suspended thread's stack frame contents are displayed in the Debug view pane, and the breakpoint line is highlighted in the editor pane in the Debug Perspective, as seen in Figure 4.7.

Whenever you run the code in debug mode and a breakpoint is encountered, the Debug Perspective appears. The behavior may be changed to remain in the Java Perspective, but the editor is positioned to the breakpoint in either perspective. See the stack trace in the Debug view. If the halt was several frames deep, you could select an earlier frame to display its variables and source code.

Select the Variables tab in the Breakpoints pane. The currently scoped variables are shown – in this case only the args variable. The lower-right pane is the Outline view. It shows the variable and method outline of the selected class. Notice the toolbar in the Debug view. Some of its icons are used to step into, stop over, and return from the current location. Hover the pointer over each icon to read its Help.

Press the right-arrow icon to resume the application, and the calculator GUI appears. Close it. The Debug view shows the terminated process. Use the pop-up menu to remove it.

#### DELETE A BREAKPOINT

You may remove a breakpoint from either the marker bar or the

Breakpoints view. Select Remove from the context menu of the marker bar, as seen in Figure 4.8, or select Remove Breakpoint from the context menu of the breakpoint in the Breakpoints view. The breakpoint will disappear from the marker bar and from the Breakpoints view.


You may delete all breakpoints by clicking the Remove All Breakpoints button in the Breakpoints view, or you may select Remove All from the view's context menu. Please know that this command removes all breakpoints in the entire WSAD workbench, not just from the active program.

The Go to File menu item in the context menu will cause the editor to display the file that has the breakpoint. The editor will position itself on the breakpoint line.

The Show Qualified Names option menu item is used to toggle the display of fully qualified package names in the breakpoint list for all breakpoints.

#### ENABLE AND DISABLE BREAKPOINTS

A breakpoint is enabled when you set it, but you may disable it from having any effect. Why would you do that? You may want to allow execution to pass through the breakpoint until the application reaches a certain state of interest. Then, you could enable the breakpoint to trap the execution the next time it passes through that point.

To disable a breakpoint, use the Breakpoints view to display all breakpoints. Right-click the desired breakpoint to see its context menu. Select the Disable item. Any disabled breakpoint displayed on the marker bar of the editor or the Breakpoints view changes from a blue or green icon to a white icon, indicating that it is not active. Enable a disabled breakpoint by using the same context menu in the breakpoints view. A disabled breakpoint's context menu shows an Enable item instead of a Disable menu item. 

# JAVAONE

[WWW.JAVA.SUN.COM/JAVAONE/SF](http://WWW.JAVA.SUN.COM/JAVAONE/SF)

*Application allows you to view management of HTTP and EJB requests*

# Manage Your Applications

BY MICHAEL J. MORTON

There are significant benefits to clustering servers in a Java 2 Enterprise Edition (J2EE) environment. Increased performance through scaling of cluster members, higher availability and reliability through automatic failover techniques, and load balancing through the use of workload management technology are the primary benefits of clustering. Given that you have deployed your application in such an environment, how do you know that your application is servicing clients in the manner that you expect for your configured environment? How can you see which server is executing a client request?



## ABOUT THE AUTHOR

Michael Morton is senior architect and an original member of the development team for IBM WebSphere Application Server Base and Network Deployment editions at IBM Research Triangle Park in North Carolina.

## E-MAIL

mortonmj@us.ibm.com

The purpose of this article is to demonstrate how a J2EE application executing in an IBM WebSphere Application Server (WAS) version 5.0 Network Deployment (ND) Edition environment can obtain runtime information about the server it is executing in. This application, referred to as the **BeenThere Workload Management Demonstration** application, is used to allow you to actually see workload management (WLM) of HTTP requests and of Enterprise JavaBean (EJB) requests. The underlying systems management infrastructure in WAS is based on the Java Management Extensions (JMX), which provide a standard framework for applications to obtain and modify runtime and system configuration, as well as pro-

viding operational control of the entire environment. It is through the use of JMX that the **BeenThere** application obtains application server runtime information programmatically.

## BeenThere Workload Management Demonstration Application

WebSphere Application Server Version 5.0 ND Edition provides workload management of HTTP requests and EJB requests. WLM of HTTP requests is done through the use of the WebSphere Web Server plug-ins. The **BeenThere** application (being used for this discussion) contains the **BeenThere** servlet, which is used to demonstrate to which application server in a cluster an HTTP request was dispatched.

Load balancing of EJB requests is managed by the WebSphere WLM Service. The **BeenThere** application contains the **Been-There** stateless session EJB used to demonstrate to which application server in a cluster an EJB request was dispatched.

Consider the configuration shown in Figure 1. First, **mjmaix** is a machine that is executing an IBM HTTP Server (IHS) that will dispatch HTTP requests to the application servers **WebServer1** and **WebServer2**, which make up the members of the cluster, **MyWebCluster**, and are executing the **BeenThere** servlet. These application servers are configured on machines **mjmnet** and **mjmthink**, respectively.

Furthermore, the WebSphere IHS plug-in has been configured such that **WebServer1** has a WLM weight of 2 and **WebServer2** has weight of 3. You can think of the weights as counters that start at their configured value and get decremented by 1 after each request the server services. Requests get dispatched in a round-robin fashion to any application server that currently has a weight counter value greater than 0. If the value is 0, it is skipped. Once all weight counters are decremented to 0, then all application server weight counters are reset to their configured value and the dispatch process starts all over again.

Then there is a **MyEJBCluster** cluster defined that is made up of the application server members **EJBServer1** and **EJBServer2**, which are responsible for executing the **BeenThere** stateless session EJB. The **BeenThere** servlet makes a call to the EJB to obtain execution information about the application server in which the EJB is executing. Furthermore, these servers have been configured with weight values of 1 and 3, respectively.

Note that this configured environment is a sample that is being used just for demonstrating the execution of the **BeenThere** Workload



Management application. In a production environment, careful resource capacity planning must be done to determine how many application server members should be created across how many machines based on the expected client request load.

Figure 2 is a screen shot of the execution of the BeenThere servlet for the configuration depicted in Figure 1. A Bean Iterations value of 7 was specified, which means the servlet made 7 invocations of the BeenThere EJB. From this, you can see the workload management execution behavior of the EJB based on the configured weight values for the cluster members of MyEJBCluster. Three invocations of the EJB execute at mjmthink for every single execution on mjmnet. Also notice that the servlet request was serviced by WebServer1 on mjmnet. Furthermore, to add a little interest to the application, Show Bean Execution Statistics was set to true to display, in milliseconds, how long it took for each EJB execution instance and the overall execution statistics.

Figure 3 is a screen shot of the use of the Show Bean Cluster Member Weights option. This displays the weight values for all the members of MyEJBCluster. Later in this article, I will discuss how the application programmatically obtains this configuration information.

Executing the BeenThere servlet a second time, as shown by Figure 4, results in IHS dispatching the HTTP request to the other member of MyWebCluster, namely WebServer2 on mjmthink. Repeated executions of the servlet will reveal a workload management behavior of the HTTP requests based on the configured weight values for the cluster members of MyWebCluster.

## Administrative Application Programming Interfaces

WebSphere Application Server version 5.0 provides administrative APIs that allow applications to access runtime and environment configuration information:

- **AdminService:** Provides the server-side interfaces to obtain application server attribute information, as well as the ability to perform standard JMX MBean management functions.

- **AdminClient:** Provides the client-side interfaces to a remote AdminService. The AdminClient class provides a proxy to the remote AdminService object through one of the supported WebSphere JMX connectors.
- **ConfigService:** The Configuration Service that provides interfaces to query or modify configuration data, locally or remotely, and manages the location and the content of WebSphere's configuration documents.

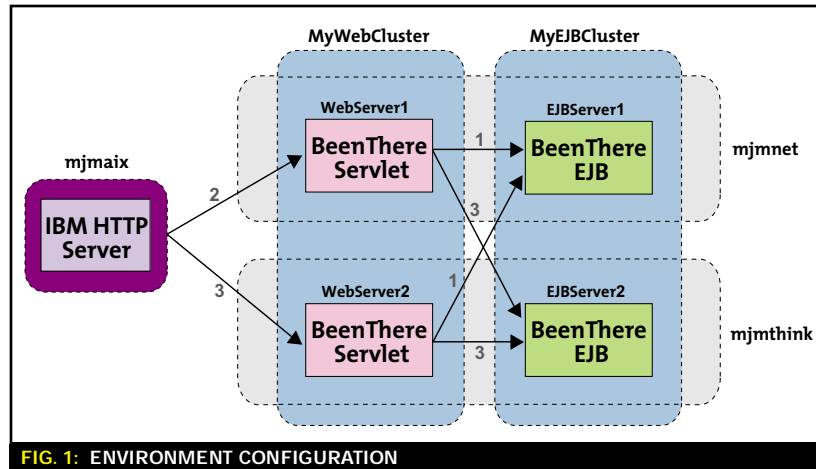


FIG. 1: ENVIRONMENT CONFIGURATION

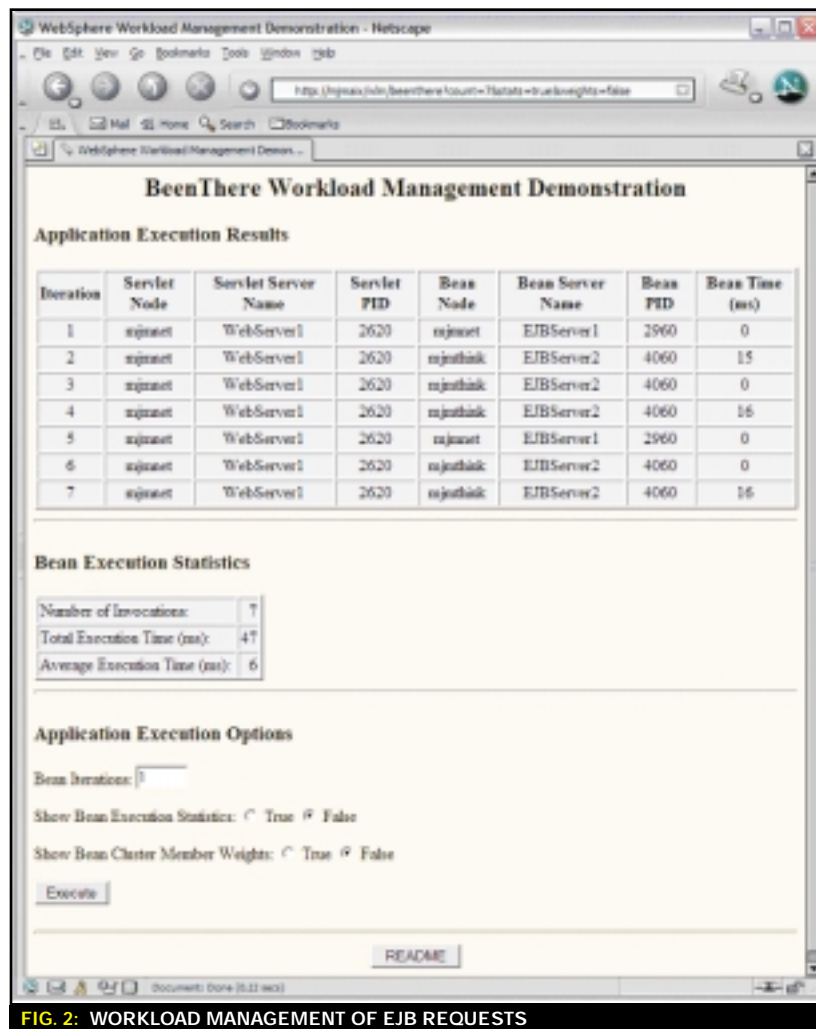


FIG. 2: WORKLOAD MANAGEMENT OF EJB REQUESTS

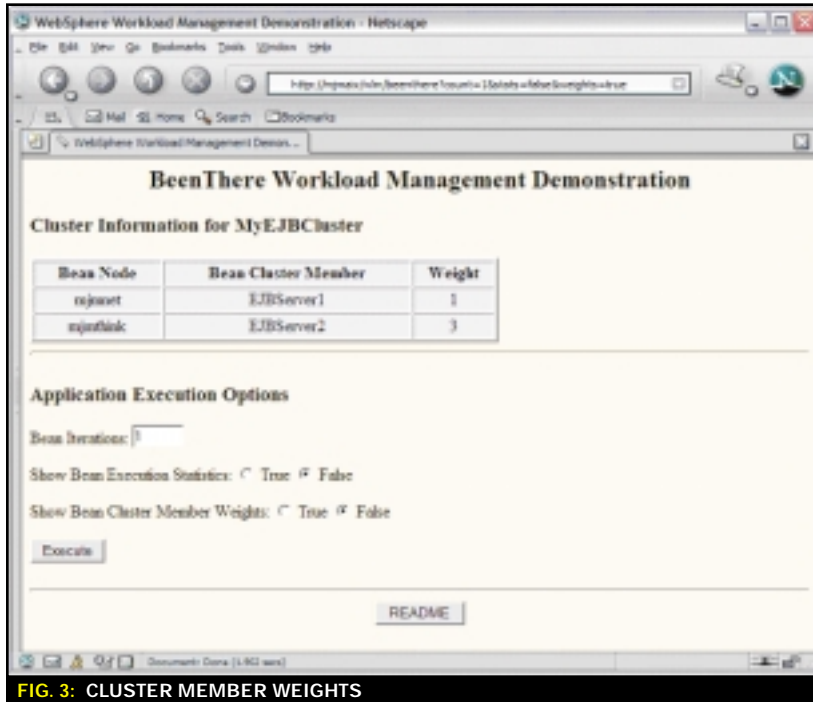


FIG. 3: CLUSTER MEMBER WEIGHTS

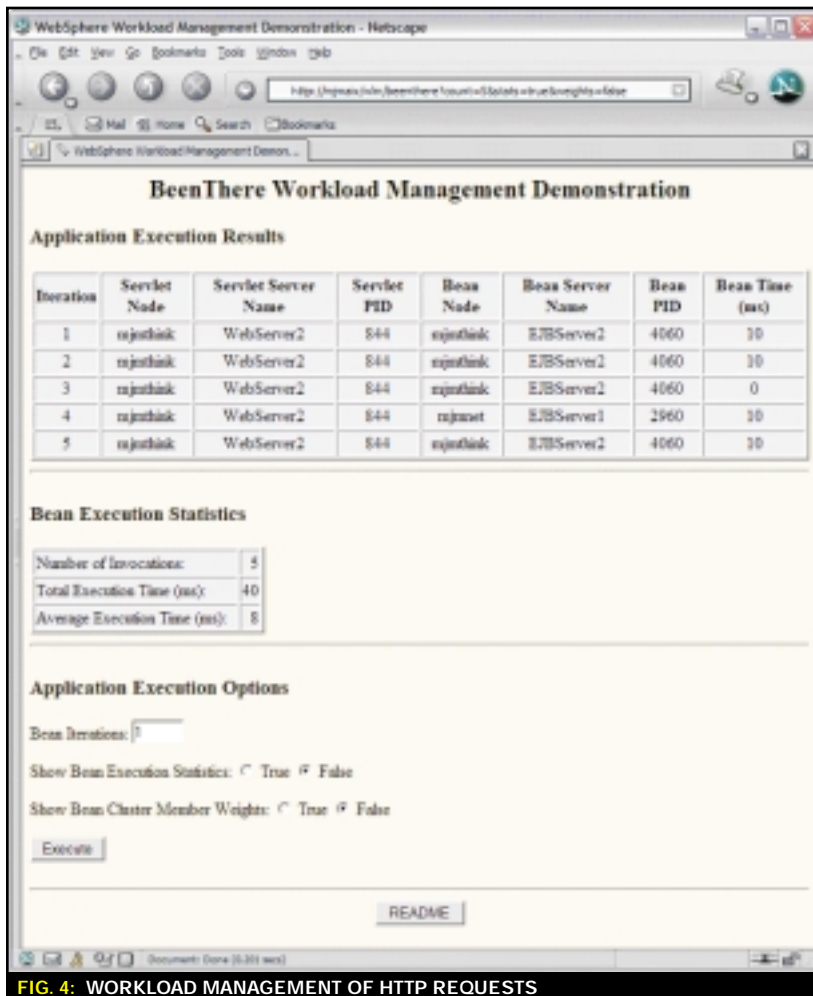


FIG. 4: WORKLOAD MANAGEMENT OF HTTP REQUESTS

In order to demonstrate the power and flexibility that applications have at their disposal when using WebSphere's administrative interfaces, the BeenThere application utilizes these interfaces in the following ways:

- **AdminService:** Both the BeenThere servlet and EJB use this interface to obtain the node name, the application server name, and the process ID for the application server they are executing in. See Listing 1 for a code segment demonstrating this.
- **AdminClient:** Used to connect to the Deployment Manager process to retrieve the Cluster Member Weight values for all the application server members of the cluster executing the BeenThere EJB. See Listing 2 for a code segment demonstrating how to obtain an instance of AdminClient.
- **ConfigService:** Used for a couple reasons. First, it is used to read the WebSphere cell.xml configuration document in order to determine if the BeenThere application is executing in a WebSphere Base Edition or Network Deployment Edition environment (see Listing 3). If an ND environment is detected, then the Show Bean Cluster Member Weights option will be enabled and displayed as an available execution option since this is only valid in this type of environment. Otherwise, this option will not be displayed as a selectable execution option. The second use of the ConfigService is to read the WebSphere serverindex.xml configuration document locally to obtain the hostname of the machine running the Deployment Manager and to obtain the port value of the SOAP connector that is used to make the JMX connection to the AdminService object of the Deployment Manager. An AdminClient instance is created using this connection information in order to retrieve the weight values of the cluster members if requested by a user.

## Configuring the WebSphere Application Server Environment

This section outlines the steps necessary to configure the environment as depicted in Figure 1, and install the

BeenThere application using the WebSphere Administrative Console. It is assumed that an IBM HTTP Server, WebSphere Network Deployment, and two WebSphere Node machines have already been installed. Furthermore, if an option on a console panel is not mentioned below, it is assumed that the default value is adequate.

The BeenThere application can be downloaded from [www.sys-con.com/websphere/sourcecfcfm](http://www.sys-con.com/websphere/sourcecfcfm).

#### CREATE THE CLUSTER MYWEBCLUSTER

1. From the console, choose Servers -> Clusters -> New.
2. In the Cluster Name dialog, specify MyWebCluster, then select Next.
3. To create new clustered servers, enter WebServer1 for the Name, then select the Node. (Select the desired machine in which to create your server. In my case I selected mjmnet.) Enter "2" for the Weight. From the Http Ports menu, choose Generate Unique Http Ports. Select Default application server template (server1) from the Select Template menu, then select Apply. For the second server, enter WebServer2 for the Name, then select the Node. (Select the desired machine in which to create your server. In my case I selected mjmthink.) Enter "3" for the Weight. From the Http Ports menu, choose Generate Unique Http Ports. Select Apply, and then select Next.
4. Select Finish, then select Save. Check the Synchronize Changes with Nodes check box. Select Save.

#### CREATE THE CLUSTER MYEJBCLUSTER

1. From the console choose Servers -> Clusters -> New.
2. Enter basic cluster information: for Cluster Name enter MyEJBCluster. Uncheck the Prefer local check box to disable node-scoped routing optimizations. (When enabled, this option specifies whether enterprise bean requests are routed to the node on which the client resides. For demonstration purposes we want the BeenThere application to exhibit workload management without enforcing this optimization so it can

be demonstrated that a servlet request to one member of MyWebCluster will dispatch an EJB request to a member of MyEJBCluster on another node.)

3. Create new clustered servers: for the name enter EJBServer1. For Select Node select the desired machine in which to create your server. In my case I selected mjmnet. For Weight enter 1. From Select Template choose Default application server template (server1). Select Apply. For the second server, enter EJBServer2 as the name. For Select Node select the desired machine in which to create your server. In my case I selected mjmthink. For Weight enter 3. Select Apply, then select Next. Select Finish, Save, and then Save again.

#### BEENTHERE APPLICATION INSTALLATION

1. From the console choose Applications -> Install New Application. To prepare for application installation you must select a local path; browse for the BeenThere50.ear file, then select Next. Leave the default virtual host name for Web modules (default\_host) selected and choose Next.
2. For Step 1, "Provide options to perform the installation," select Next.
3. For Step 2, "Provide JNDI Names for Beans," select Next.
4. For Step 3, "Map EJB references to Beans," select Next.
5. For Step 4, "Map virtual hosts for web modules," select Next.
6. For Step 5, "Map the modules to application servers," first select the cluster, MyWebCluster. Check the check box for the BeenThere WAR. Select Apply. Now select the cluster, MyEJBCluster. Check the check box for the BeenThere EJB. Select Apply, and then select Next.
7. For Step 6, "Ensure all unprotected 2.0 methods have the correct level of protection," select Next, then Finish.
8. Select Save to Master Configuration, then select Save.

#### UPDATE THE VIRTUAL HOST

Each application server has its own HTTP transport that receives and responds to HTTP requests. Therefore

they must be configured to use a unique hostname and port value for communication. During the creation of the cluster MyWebCluster, the option "Generate Unique Http Ports" was selected for each new cluster member. This option instructs the WebSphere administration to assist with avoiding HTTP port conflicts by creating a unique port value for each of the new application servers being created. It is through the use of a Virtual Host that a single machine can appear to resemble multiple host machines.

Each virtual host has a logical name and a list of associated aliases where a host alias consists of a TCP/IP hostname and port value that define the HTTP transport channel to be used for an application configured for the logical virtual host. As is the case with the BeenThere application, it has been configured to use the virtual host default\_host, therefore you must ensure that the HTTP port value dynamically created has an associated host alias entry configured for the default\_host virtual host.

1. Still using the WebSphere Administrative Console, determine the HTTP port values that were dynamically created: choose Servers -> Application Servers -> WebServer1 -> Web Container -> Http Transports.
2. Take note of the Host and Port values and now perform the following to configure a host alias entry for the default\_host virtual host: choose Environment -> Virtual Hosts -> default\_host -> Host Aliases.
3. Create a new Host Alias entry using the host and port values determined for WebServer1.
4. Repeat this procedure for WebServer2 to ensure default\_host is now correctly configured.

#### GENERATE THE HTTP SERVER PLUG-IN FILE

1. Choose Environment -> Update WebServer Plugin.
2. You have two choices here:
  - For this scenario, selecting OK will create the plugin.cfg.xml file and place it in the directory <WAS\_ROOT>/config/cells on

the machine running the Network Deployment Manager. This is convenient if you are also running your HTTP Server on the same machine, since the HTTP Server will detect that the plugin-cfg.xml file has been updated and automatically reload it. So if your server is configured to refer to this location, it will dynamically reload the new configuration every time you select OK.

- Alternatively, if you are running your HTTP Server on a separate machine, you can select View to then perform a save from the browser to save the plugin-cfg.xml file to the machine on which you are using your browser. In my case, I am running the IBM HTTP Server on the machine mjmaix, so using a browser on that machine, I use this method to save a new plugin-cfg.xml file.

#### ENABLING THE WEBSPPHERE CONFIGURATION SERVICE

The WebSphere Configuration Service by default is not enabled for

application servers. As mentioned previously, the BeenThere application requires this service to have the ability to programmatically read WebSphere configuration files to obtain environment information. Perform the following steps in the WebSphere Administrative Console to enable this service:

1. Choose Servers -> Application Servers -> WebServer1 -> Administration Services -> Custom Properties -> New.
2. In the Name dialog box, enter "com.ibm.websphere.management.enableConfigMBean". Select Next.
3. In the Value dialog box, enter "true". Select Apply, then select Save, and again select Save.
4. Repeat this procedure for WebServer2.

#### START YOUR SERVERS!


From the console, select Servers -> Clusters. Check the MyWebCluster and MyEJBCluster check boxes. Then select Start.

In the IBM HTTP Server configuration file httpd.conf, ensure it refers to the correct location of the newly

generated plugin-cfg.xml file and start the HTTP Server if it is not already started. If it is already started, it will automatically detect that the plugin-cfg.xml file has been updated and reload it.

Congratulations! You should now be able to see workload management in action by entering the URL `http://<servername>/wlm/been-there`, where servername is the name of the machine running your IBM HTTP Server.

#### Conclusion

WebSphere Application Server Workload Management technology provides an enterprise-level platform for J2EE applications that demand a scalable, highly available environment. Verifying that your application is executing in a manner that is expected for your configuration can easily be done by utilizing WebSphere's administrative APIs, as was shown through the use of the BeenThere demonstration application. This application allowed you to see workload management of HTTP and EJB requests in action. 

#### LISTING 1

```
// Get the WebSphere AdminService.
adminService = AdminServiceFactory.getAdminService();

// Get the WebSphere Admin Local Server MBean instance.
localServer = adminService.getLocalServer();

// Get the Node name.
String nodeName = (String)
adminService.getAttribute(localServer, "nodeName");
beenThereServletBean.setServletServerNodeName(nodeName);

// Get the Application Server name.
String serverName = (String)
adminService.getAttribute(localServer, "name");
beenThereServletBean.setServletServerName(serverName);

// Get the Application Server Process Id.
String serverPid = (String)
adminService.getAttribute(localServer, "pid");
beenThereServletBean.setServletServerProcessId(serverPid);
```

#### LISTING 2

```
Properties props = new Properties();
props.setProperty(AdminClient.CONNECTOR_TYPE,
    AdminClient.CONNECTOR_TYPE_SOAP);
props.setProperty(AdminClient.CONNECTOR_HOST, host);
props.setProperty(AdminClient.CONNECTOR_PORT, port);
AdminClient adminClient =
    AdminClientFactory.createAdminClient(props);
```

#### LISTING 3

```
// Create a new WebSphere Management Session.
Session session = new Session();

// Get the WebSphere ConfigService instance for this
application server executing this servlet.
ConfigService configService =
    ConfigServiceFactory.getConfigService();
if (configService != null)
{
    // Read the cell.xml document.
    ObjectName cellObj =
        ConfigServiceHelper.createObjectName(null, "Cell");
    ObjectName[] cellObjs = configService.queryConfigObjects
        (session, null, cellObj, null);

    if (cellObjs.length != 0)
    {
        cellObj = cellObjs[0];
        String cellName = (String) configService.getAttribute
            (session, cellObj, "name");
        String cellType = (String) configService.getAttribute
            (session, cellObj, "cellType");
        if (cellType.equals("DISTRIBUTED"))
        {
            websphereEND = true;
        }
    }

    // Release the Session.
    configService.discard(session);
}
```

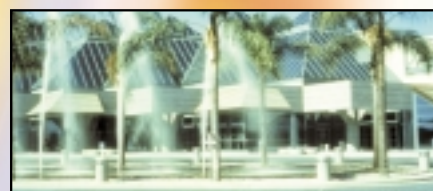


# Web Services Edge <sup>WEST</sup> 2003

**web services**  
conference & expo **EDGE**

SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



EXTENDING THE ENTERPRISE  
WITH WEB SERVICES THROUGH JAVA,  
.NET, WEBSphere, MAC OS X  
AND XML TECHNOLOGIES



XML

WebSphere



- Featured technologies and topics will include:
- Focus on .NET
  - Focus on Java
  - Focus on WebSphere
  - Focus on Mac OS X
  - Focus on XML

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



Over 100 participating companies will display and demonstrate over 300 developer products and solutions.

Over 2,000 Systems Integrators, System Architects, Developers, and Project Managers will attend the conference expo.

Over 60 of the latest sessions on training, certifications, seminars, case studies, and panel discussions will deliver REAL World benefits, the industry pulse and proven strategies.

Contact information: U.S. Events: 201 802-3069 or e-mail [grisha@sys-con.com](mailto:grisha@sys-con.com)

WebServices JOURNAL

JAVA DEVELOPER'S JOURNAL

WebSphere DEVELOPER'S JOURNAL

XML JOURNAL

.NET JOURNAL

SAMS

WebLogic

wireless BUSINESS TECHNOLOGY

LINUX BUSINESS TECHNOLOGY

CE Advisor

COLOFUSION Developer's Journal

PowerBuilder Developer's Journal

Java is a registered trademark of Sun Microsystems. .NET is a registered trademark of Microsoft. Mac OS X is a registered trademark of Apple Computer, Inc. WebSphere is a registered trademark of IBM. All other product names herein are the properties of their respective companies.

**WEB SERVICES EDGE WEST 2003**  
**CALL FOR PAPERS NOW OPEN**  
Submit your papers online at:  
[www.sys-con.com/webservices2003west](http://www.sys-con.com/webservices2003west)

PRODUCED BY  
**SYS-CON**  
EVENTS

*Redefined CMP component model offers better performance and flexibility*

# Step-By-Step EJB 2.0 CMP/CMR Development

BY SWARAJ PAL, DEEPPENDRA SHRESTHA, AND RAGHU SHRESTHA



## ABOUT THE AUTHOR

Swaraj Pal is a senior WebSphere consultant at Noospherics Technologies, Inc. He has seven years of IT experience and has worked with WebSphere for several years. Swaraj writes articles on the WebSphere suite of products for various magazines and white papers on WebSphere performance and tuning using the Mercury suite of products.

## E-MAIL

swarajpal@yahoo.com

The EJB 2.0 specification enhances the Enterprise JavaBeans paradigm in many ways. The entire concept of the container-managed persistence (CMP) component model has been redefined for better performance and flexibility. To make your work a lot easier, IBM quickly introduced WebSphere Application Developer (WSAD) 5.0, which supports both the EJB 1.1 and 2.0 specifications. In this article we will discuss some important concepts that will provide you a good basis for working with the new specification.

## CMP Component Model

Persistence service, usually provided by the container, has been redesigned with an entirely new CMP component model. Unlike with the EJB 1.1 container, EJB 2.0 persistence is handled by a new entity called the “persistence manager.” This makes the persistence mechanism independent of the container provider, thus making CMP entity beans portable to any EJB 2.0-complaint container, irrespective of the vendor and back-end relational databases. The persistence manager also handles the relationships among all of the CMP entity beans as well. To make this work, the CMP

entity beans are introduced with abstract class qualifiers and all the persistence and relationship fields are accessed using abstract getters and setters, which is known as “abstract persistence schema,” for example:

```
public abstract Customer
implements
javax.ejb.EntityBean{
    //
    public abstract void
    setPhone(){}
    public abstract String
    getPhone(){}
    //
}
```

During deployment, the persistence manager uses the persistence information in the EJB deployment descriptor (DD), the bean's persistence schema, to generate concrete bean classes, which are instantiated at runtime.

## Local Interface vs Remote Interface

EJB 2.0 introduces local interfaces, which either coexist with the remote interface or exist alone. Prior to creating EJBs it is wise to decide on the design aspects of an entity EJB. The introduction of local interfaces has improved performance, but it makes the job of the designers more challenging since they need to decide whether to use only a local interface or both a local and remote interface. If an EJB has only a local interface, then clients within the same JVM can access the EJB, but remote clients will not see it. On the one hand this is good, since you are adding pseudo-security by not allowing outside clients to access your EJB. But remember that for a distributed Web application or Web service where you are allowing clients outside the JVM to access the EJB, you need to add remote interfaces. WSAD 5.0 gives you the option of adding local and remote interfaces to your EJB (see Figure 1). So if you want to test your EJBs using a simple Java client, make sure to add the remote interface to your EJBs or you won't be able to work with them.

## Container-Managed Relationships

Container-managed relationships (known as CMR) are now standardized in the EJB 2.0 specification. WSAD lets you create relationships between the CMP entity beans. It supports all three types of relationships between data: one-to-one, one-to-many, and many-to-many.

To clarify the entity relationships, we will exercise the steps to create

two entities using WSAD and to create a one-to-many relationship between the Customer and Order CMP entity beans. For our example we used DB2 7.2 Fixpack1 as our relational database. Before we start, we want to choose the approach for mapping our entities to the respective tables. There are three standard approaches in creating and persisting field mapping: Bottom Up, Top Down, and Meet in the Middle.

You would use the Bottom Up approach for creating entities that represent an existing relational database. In this case you could use the WSAD wizard to magically create all the entities and their relationships. With the Top Down approach you would create the entities using the Create Enterprise Beans wizard and then you would create the EJB-to-RDB mapping. You could ask WSAD to generate the data definition language file for the tables, which will be used as a persistence store by the beans. On the other hand, the Meet in the Middle approach is used for handshaking between already created database tables and entity beans.

#### CREATE TWO CMP ENTITY EJBS

- Create a J2EE 1.3 Enterprise Application project "OnlineStore" with the EJB module "OnlineStoreEJB". Make sure to uncheck the Application Client and Web Modules check boxes. We don't need them for our example.
- Create the entity EJBS: from the context menu select File->New->Enterprise Bean->. This will take you to the EJB Creation wizard.
- On the first page, select the "OnlineStoreEJB" EJB Module from the list and select Next.
- On the Create a 2.0 Enterprise Bean page, select CMP 2.0 Bean. Provide Customer as the bean name (the first letter of the bean name should be in caps, by convention). Set com.store.ejb as the Default Package and select Next.
- On this Enterprise Bean detail page make sure to select Local Client View and deselect Remote Client View (see Figure 1).
- Now we need to add persistence fields to the bean. We will add three CMP attributes to the Customer bean: cust\_id as type

Java String (also the Primary Key Field); fname, also as a Java String; and lname, as a Java String. Add these attributes in the CMP Attributes text area. Click Finish. This will create the abstract Customer bean class and the two local interfaces.

- Follow the same steps to create the Order entity. For order, add the CMP attributes order\_id (Java String and Primary Key Field); date (sql date); amount (double), desc (Java String).

#### CREATE A ONE-TO-MANY RELATIONSHIP

- Open the DD Editor on the Overview tab, and find the group name "Relationships 2.0".
- Click on the Add button under the Relationships heading to open the Add Relationship wizard.
- On the first screen you can see all of your entity beans on both sides. Select one bean from the left side (Customer) and another bean from the right (Order). The name for the relationship "Customer-Order" is automatically set. Click Next.
- WSAD assigns relationship roles for your beans. In the Relationship Roles page, under the Customer bean, set a role named "orders". Multiplicity should be set to One. For the Order bean, set the multiplicity to Many. You'll see that the field type under Customer is now java.util.Collection. You should check the Cascade Delete check box under Order to ensure that all orders are deleted if a customer is deleted. Now set the navigable check box on both sides as checked. You'll also see that the Foreign Key check box is checked but disabled. This is because for a one-to-many relationship a foreign key is mandatory. Click Finish. This will append the getOrders() and setOrders (Collection) methods to the Customer bean and promote these methods to the CustomerLocal Interface. Also in the OrderBean class and the OrderLocal interface you can see two generated methods named

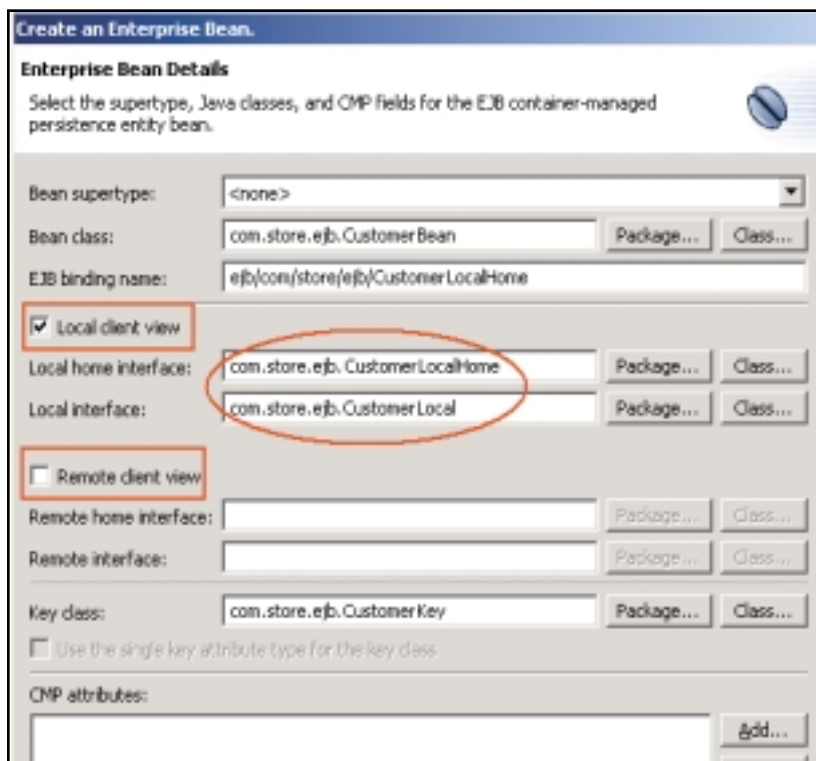


FIG. 1: WSAD 5.0 EJB 2.0 CMP BEAN CREATION WIZARD



#### ABOUT THE AUTHOR

Deependra Shrestha is a Sun Certified Java programmer and IBM Certified WebSphere developer with extensive experience in analysis, design, development, and administration of WebSphere and J2EE applications. He is a senior programmer for Client Network Services, Inc. Deependra has been involved in a number of WebSphere and J2EE projects for various large organizations.

#### E-MAIL

deependra202@yahoo.com





#### ABOUT THE AUTHOR

Raghu Shrestha is a Sun Certified J2EE Programmer and an IBM Certified WebSphere developer with comprehensive experience on application server technologies. He worked extensively as a WebSphere v5 release testing specialist. He has extensive analysis, design, development, and administration skills in these technologies.

#### E-MAIL

raghu@noospherics.com

getCustomer and setCustomer (CustomerLocal). Save the DD. *Note:* Notice that the CMR relationships generated methods only for the local interfaces. You have to code your own methods to add this in the Remote interface.

#### CREATE THE EJB-TO-RDB MAPPINGS

Once the relationships are generated we need to map the relationships to the database side.

- Switch to the J2EE perspective and select the J2EE hierarchy tab.
- Right-click on the EJB module. Select Generate->EJB to RDB mapping. This will open the EJB to RDB Mapping wizard.
- On the first page select "Create a new backend folder" and click Next.
- "Create new EJB/RDB Mapping" will give you two options for creating the mapping: Top Down and Meet in the Middle. (An explanation of each is given on that page.) In our case we will be creating the mapping as Top Down. So select Top Down and click Next.
- On this page provide the database vendor, database name, and schema name. In our example they will be DB2 UDB 7.2, onlinest, and administrator respectively. Make sure to create the onlinest database separately in your DB2 installation. Also, make sure that you have selected the "Generate DDL" check box, and click Finish. This will create the mapping and WSAD will open the mapping editor (Map.mapxml) automatically for you (see Figure 2). *Note:* We noticed one limitation in the WSAD-generated DDL. The foreign key relationship definition is not generated. Make sure you manually add the foreign key constraint into the database tables created by the DDL.
- Select table.ddl from the J2EE navigator panel under META-INF and right-click Run on Database Server. Make sure to create a new connection,

"onlinest", which connects to the onlinest database.

#### Configure the WebSphere Test Environment

- Switch to the Server perspective. Under the Server Configuration Panel, right-click on Servers and select New->Select Server and Server Configuration. This will open the New Server configuration wizard.
- Provide a server name, "onlinest", and click Finish. This will create a new WebSphere test server, where we will test our EJBs. WSAD will automatically open the WebSphere Server Configuration editor.
- In the editor switch to the Datasource tab. From the JDBC Provider list select Default DB2 JDBC Provider and click Edit.
- On the Edit page, you need to provide the path of the db2java.zip file on your file system. Remove the default entry and add the actual db2java.zip by clicking the Add External JARs button. Then click Finish to return to the editor.

- Now we need to add a datasource to the onlinest database using the driver we just configured. So click Add under Data Source. In the Datasource wizard click Next to skip the first page by accepting the default values. On the second page you should notice that by default WSAD generated a JNDI name for the datasource; you can accept the default or enter your own JNDI name. We will accept the default, jdbc/ds1 (because this is the first datasource). Click next. Remember this datasource name. We will need this later.
- On the Modify Resource Properties page we need to specify the database name. Provide the database name as the value of databaseName property. Click Finish. Save the server configuration.
- Now go back to the DD page or open it. On the Overview tab, scroll all the way to the bottom. Under JNDI-CMP Factory Connection Binding, JNDI name, specify the datasource JNDI name we just created (jdbc/ds1). Save the DD.

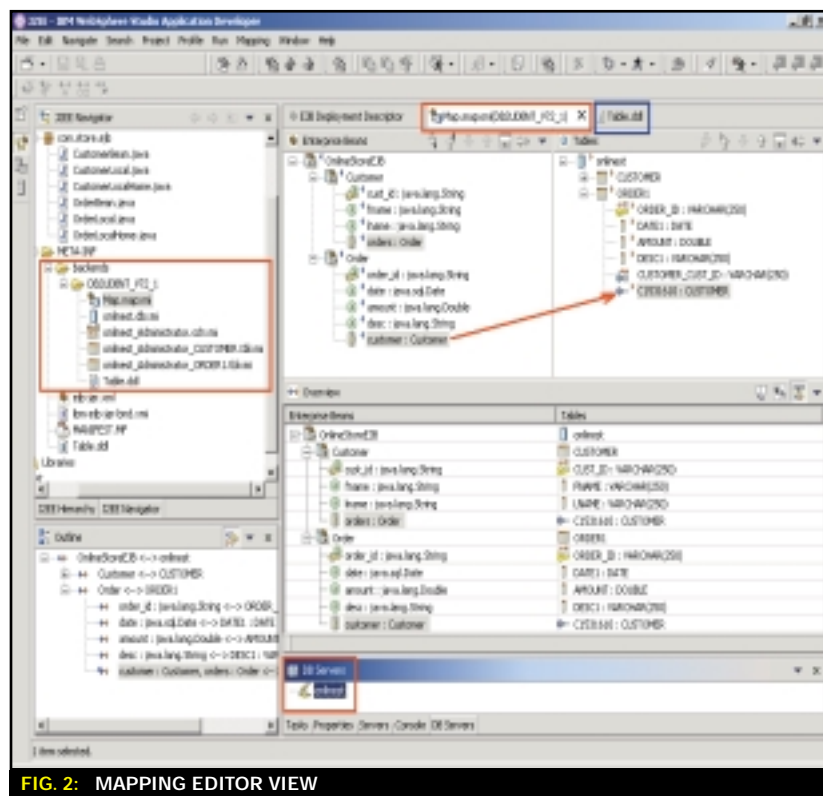


FIG. 2: MAPPING EDITOR VIEW



Now that we have created two entities and established a relationship between them, we are ready to test our EJBs and their relationships. The EJB test client WSAD provides is not the best tool to test the relationships, since each method call from the test client is a transaction. So when you invoke `getOrders()` in `Customer`, it returns a `Collection` object that gets invalidated as soon the transaction ends and when you try to work with the `Collection` object, you will encounter an `IllegalStateException` for the object. We encourage you to test your entity beans using a session facade to access them. Let's build a quick session facade for the `Customer` entity bean using its local interface.

### Create a Session Facade as the Entity Client

Create a `Customer Facade` stateless session bean, `CMRTester`, using the `Create Enterprise Bean` wizard. After WSAD creates the bean skeleton, open the session bean class and add a method called `doTesting()`. Add the code shown in Listing 1 to access the `Customer` bean. Save the bean. Switch/Open `EJB Deployment Descriptor`. Now we need to create a local `Customer`

reference under the session bean, so that the session bean can look up the `localhome` interface for the `Customer` entity. Open DD, click on the `References` tab, select the session bean, and click `Add`. Select `EJB Local Reference`. In the `EJB Local Reference` page, under `Link`, click `Browse` to select the `Customer` bean under `Current Enterprise Project`. This action will create a local reference (`ejb/Customer`) of the `customer` bean under the session bean.

### Test the Entity Beans Using the EJB Test Client


Switch to the `J2EE` perspective and select the `J2EE Hierarchy` tab. Expand `EJB Modules`. Expand the `OnlineStoreEJB` module. Right-click the `CMRTester` session bean and select `Run On Server`. You will be given the option to run the bean on an existing server. The created server will automatically be selected. Click `Next`. On the second page make sure that the `Deploy EJB Beans` check box is selected. This will generate the `RMIC` code and the concrete implementation of the abstract bean class, which we discussed earlier. You could also generate the class by right-clicking the `EJB` module and selecting `Generate`

->Deploy and `RMIC` code... This will generate the classes necessary to run the beans on the test server. This action will also publish the EJBs to the test server. After successful startup of the server, the `EJB Test Client` will automatically start up. Now you can test your EJBs using the test client. Add some data to the tables to test.

### March Ahead...

Thus far we've passed on some insights into the new enhancements in the `EJB 2.0` specification. You can see that the abstract persistence schema is the basis of the new persistence mechanism and relationships. The specification also introduces message-driven beans and defines a query language, known as `EJB-QL`, to define the finder and select methods in the `CMP` entity EJBs. This specification surely comes with lot of promises that require more exploration. We hope you are now more enthusiastic about exploring in-depth. Happy EJBing.

### Acknowledgment

The authors would like to thank Ashim Ranjitkar, Raju Mukherjee, and Soutik Singha for their contributions to this article. 

#### LISTING 1

```
//import Statements
public class MyCMRtesterBean implements
javax.ejb.SessionBean {
    private javax.ejb.SessionContext mySessionCtx;
    //getSessionContext
    //setSessionContext
    //ejbCreate

    public void ejbCreate() throws javax.ejb.CreateException {
        try{
            InitialContext ctx = new
InitialContext();
            cHome =
(CustomerLocalHome)ctx.lookup("java:comp/env/ejb/Customer");
        }catch(Exception e){System.out.println(e);}
    }
    //ejbActivate
    //ejbPassivate
    //ejbRemove
```

```
public void doTesting(){
    try{
        CustomerLocal cBean =
(CustomerLocal)cHome.findByPrimaryKey("0001");
        Collection orders = cBean.getOrders();
        Iterator it = orders.iterator();

        OrderLocal order = null;
        while(it.hasNext()){
            order = (OrderLocal)it.next();
            System.out.println("Order id =
"+order.getOrder_id());
            System.out.println("Date =
"+order.getDate());
            System.out.println("Amount =
"+order.getAmount());
        }
    }catch(Exception e){System.out.println(e);}
}
```

*From M7*

# M7 Application Assembly Suite

BY JAY JOHNSON

Those who have battled J2EE application development with the plethora of WebSphere tools have probably thought that there must be an easier way. Now there is a new breed of software called application assembly platforms (AAP). These toolsets make it possible to visually assemble an enterprise application without getting bogged down in the details. The components can come from multiple sources. The best of these tools are server independent or at least support the "Big 2" application servers – WebSphere and WebLogic. The M7 Application Assembly Suite is one such tool.

## Product Info:

**COMPANY:**  
M7 Corporation  
10101 N. De Anza Blvd.  
1st Floor  
Cupertino, CA 95014  
E-mail: sales@m7.com  
Web: www.m7.com  
Telephone: 408-850-0700

**SYSTEM REQUIREMENTS:**  
Supports all standard  
J2EE application servers,  
including IBM  
WebSphere, BEA  
WebLogic, and JBoss



## ABOUT THE AUTHOR

Jay Johnson is an independent J2EE architect.

## E-MAIL

jay2ee@hotmail.com

It includes a graphical flow builder, a component repository, a JSP graphical editor, and a persistence engine. It is designed to work in concert with a component-creation IDE such as WebSphere Studio Application Developer or JBuilder. The M7 Application Assembly Suite is focused on helping application programmers put components together graphically in a workflow and creating a servlet/JSP presentation layer.

Enterprise applications have one major weakness: for the less experienced programmer, it's very difficult to put all the pieces together to make them. Creating the individual components is challenging, but assembling them into an application can be even more difficult. Most of the problems spring from the fact that developers must create

and use components at the same time. There is no clear division of responsibility.

In the excitement to move to the J2EE architecture, particularly in regard to the clients I've worked with, there has been a breakdown in the differentiation of roles in a J2EE project. For example, one mistake is to confuse J2EE component users with component creators. In most projects, a team of J2EE architects and developers should be in charge of creating the needed components, while a team of application programmers is in charge of using those components to fulfill user requirements. This, however, is usually not the case.

The customers I've worked with, especially in the past year, have existing teams of COBOL and/or Visual Basic programmers. Their project

managers hope they can be turned into J2EE component developers. The reality is that most of these application programmers cannot get up to speed in J2EE in time to benefit a project, and in fact often have a negative impact on the schedule. This forces J2EE projects to turn to expensive component-development talent, or to cancel the project. This also means an IT organization will not spend its budget to build up its in-house resources.

WebSphere Studio is a great component-development tool suite that can greatly expand the productivity of architects and developers already experienced in J2EE. It is overwhelming, however, for the average application programmer.

The M7 Application Assembly Suite is version 3.x software, so it is relatively smooth and bug free. It runs in conjunction with an application server, either WebSphere, WebLogic, or JBoss. Rather than providing a proprietary application server, the product offers its own interface layer, which executes between the application and the application server, making development, testing, and deployment easier. This layer facilitates the M7 Application Assembly Suite's visual workflow builder, JSP editor, and persistence engine as well. The runtime portion of the product is really a JAR file that gets deployed with the application. This is the runtime support for workflow, JSP tags, and the persistence engine. It's not a container, and in fact requires an EJB container to execute, just like any J2EE application.

Many projects require programmers to solve too many problems at once. AAPs like the M7 Application Assembly Suite make it easier to solve one problem at a time. They also make it easier to split the problems up between programmers and to integrate the individual components. Take the Model/View/Controller architecture. Ideally, the tiers will be very loosely coupled so development can be done independently.

Architects can work on creating components and interfaces, while developers work on the business logic, and application programmers work on the view. It is reasonable to get the view tier working independently of the model and the controller, if the component interfaces are defined early on. This means no single team member has to know how to do everything – each can specialize in his or her own part of the project.

Almost every serious J2EE project involves EJBs, but introducing EJB development and EJB container technology into the application programming effort can bring progress to a full stop. EJB technology is a quantum leap for many application programmers, since most have experience only in VB or in a mainframe environment. EJBs require experience in an alphabet soup of technologies including JTA, CMP, JNDI, MDB, BMP, etc.

It is an old maxim that any problem in computer science can be solved by adding another level of indirection. With the M7 Application Assembly Suite, programmers do not directly edit Java components, but rather XML representations that configure the components. This means that code can be changed and run with no need to rebuild or reassemble Java components.

Ironically, the view tier of an enterprise application presents the greatest challenge for reuse, with embedded code for accessing components in the presentation layer and the control layer. The M7 Application Assembly Suite abstracts out the details, and provides a graphical flow builder (see Figure 1) and a repository that can be viewed from multiple perspectives in a way similar to Eclipse and WebSphere Studio. The repository is mapped directly to a source management tool such as CVS or ClearCase. The repository can be shared by multiple departments/ projects, and holds a variety of components, including EJBs, JavaBeans, JSPs, etc.

The persistence layer in the M7 Application Assembly Suite does dynamic mapping between any SQL database and the Java objects an application uses, without generating Java code. When a programmer imports a schema as a data object entity in the

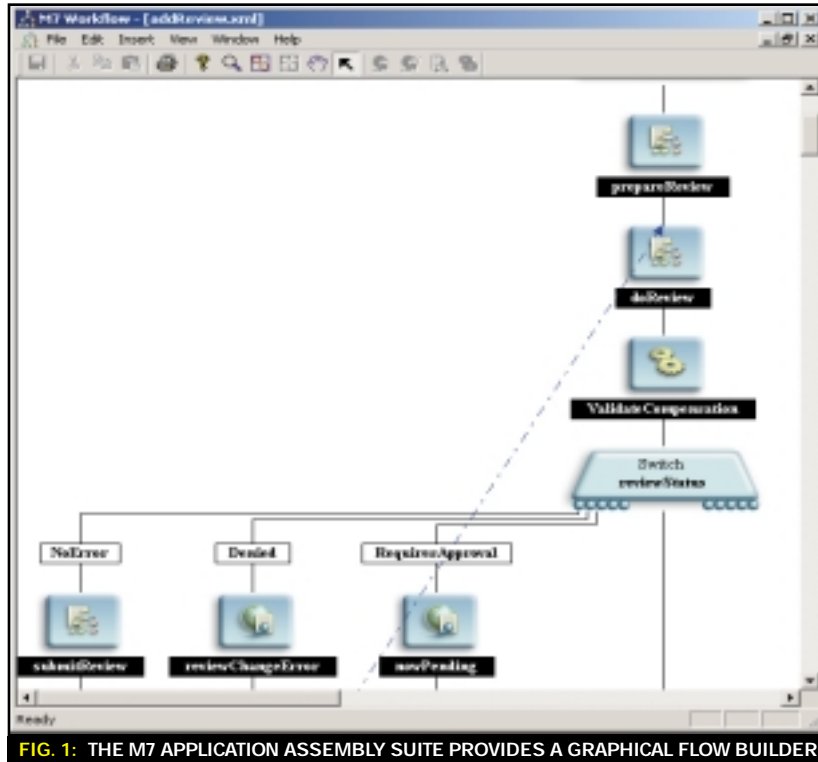


FIG. 1: THE M7 APPLICATION ASSEMBLY SUITE PROVIDES A GRAPHICAL FLOW BUILDER

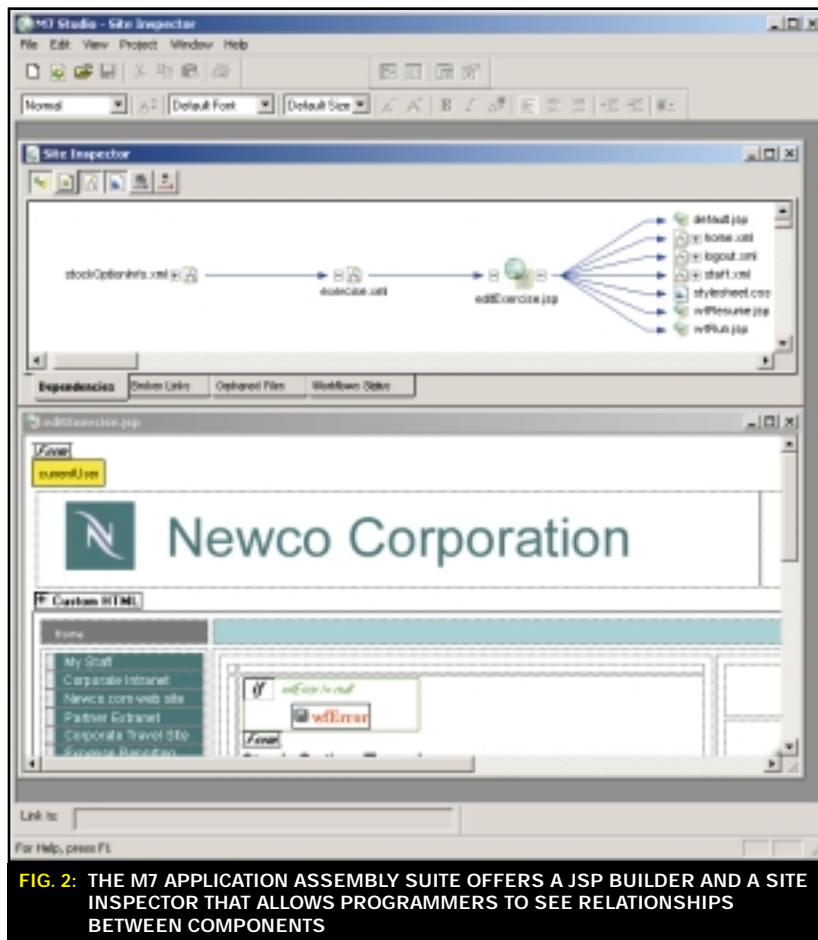


FIG. 2: THE M7 APPLICATION ASSEMBLY SUITE OFFERS A JSP BUILDER AND A SITE INSPECTOR THAT ALLOWS PROGRAMMERS TO SEE RELATIONSHIPS BETWEEN COMPONENTS

-continued on page 45

*Enjoy the same advantages as static-content publishers*

# Accelerating Delivery of Personalized Web Content

BY STEVE IMS

Web-caching technologies offer great advantages to static-content publishers, including improved (faster) response times for end users and reduced data center workload. Further, because these technologies are based upon an open standard (HTTP), content publishers have a choice of solution providers, including hardware-based and software-based solutions, as well as service offerings on commercial content-distribution networks, such as Akamai Technologies' EdgeSuite.

Unfortunately, traditional caching technologies offer minimal benefit to personalized Web content (because caching provides more benefit to content that's shareable by multiple users). As a result, application servers are always required to handle requests for personalized content.

Thus, personalized content, which is vital for competitive e-businesses, is "expensive" to generate. This cost may be realized in higher infrastructure costs (that is, more application servers and greater bandwidth demands) as well as increased end-user response times.

Distributed fragment caching and assembly may change all that. How? By redefining the cacheable units from whole pages to smaller "fragments" of pages. A Web page that contains any user-specific personalization is not shareable and thus not worth caching as a whole. However, most personalized pages can be modeled as aggregations of smaller content fragments. (For example, Listing 1 (available at [www.sys-con.com/websphere/sourcec.cfm](http://www.sys-con.com/websphere/sourcec.cfm))

shows a personalized page, `home.jsp`, that has been divided into fragments using WESI include tags. Each fragment uses the WESI surrogateControl tag to indicate whether or not it can be cached and for how long. Here the WESI surrogateControl tag in `computers.jsp` indicates that it can be cached for a maximum of 300 seconds.)

If many of these fragments are shareable, then you can realize benefits by caching the fragments and assembling them on demand, along with retrieved noncacheable fragments, into personalized pages. This is the basis of a technology called Edge Side Includes.

Edge Side Includes (ESI), which is a W3C Acknowledged Submission ([www.w3.org/Submission/2001/09](http://www.w3.org/Submission/2001/09)), defines an extension of traditional Web caching that enables a Web-caching proxy to cache and assemble content fragments into personalized Web pages. ESI, which is based upon a two-tier architecture (proxy and origin server), describes three basic elements:

1. A "markup language," embedded in responses from the origin server,


- that identifies fragments and instructions for fragment assembly
2. A means by which the proxy and origin server can communicate regarding their capabilities and intentions
3. A means to invalidate cached fragments on demand

Although ESI holds great promise for personalized Web content, there is still a chasm for application developers: ESI is not integrated with any mainstream programming model. That is, ESI relies on a set of unique tags (markup language) and HTTP headers. This, unfortunately, makes the "cost" of ESI exploitation rather high.

IBM's WebSphere team has introduced a new technology on alphaWorks that bridges this chasm: WebSphere APIs for ESI (WESI). WESI is a set of Java APIs and JavaServer Pages custom tags that give WebSphere application developers direct control over ESI functionality. Further, WESI supports integrated testing within WebSphere Studio. Thus, WESI enables developers to rapidly develop, test, debug, and deploy ESI-enabled applications.

WESI also promotes interoperability. Since WESI is based on the published ESI specifications, WESI users have their choice of ESI-enabled proxies, including products from WebSphere and service offerings from Akamai Technologies.

Finally, WESI makes it easier for WebSphere customers to enjoy the advantages that traditional caching brought to static-content publishers. For one, redundant workload required to generate personalized Web pages can be offloaded from application servers to less-expensive proxies. Also, end-user response times may be improved (reduced) since proxies are typically closer, in terms of network latency, to the end users.

The bottom line is that WESI enables WebSphere customers to accelerate the delivery of personalized Web content. 



## ABOUT THE AUTHOR

Steve Ims is a senior software engineer for IBM's WebSphere Advanced Design and Technology group. He focuses on extending WebSphere runtime services "into the network," along with software-development tools that enable rapid exploitation of these services.

## E-MAIL

[steveims@us.ibm.com](mailto:steveims@us.ibm.com)



M7 repository, an XML descriptor gets created that describes everything needed by the persistence engine to do its work at runtime. There are a couple of benefits to the persistence layer: programmers can quickly start building applications with direct access to the database, and the persistence layer implements a Value-Object pattern (basically caching for reduced network traffic) for enhanced performance. This means that entity beans or some other persistence strategy can be added in later in the project as necessary.

Developing an application with the M7 Application Assembly Suite is like deploying an exploded EAR/WAR file. There's a notion of a deployment set that can include the app under development. The M7 Application Assembly Suite will then tell the app server where the deployment directory is and handle all of the deployment nuances of each app server. When the developer is ready to actually put an app into production, he or she JARs up the app directory tree and puts it in the appropriate deployment directory for the app server.

A J2EE developer could use WebSphere Studio to create a component consisting of a session bean and several entity beans, for example, and place it in the repository. Using the repository, the developer can control how much of the component is exposed to the application programmer. The programmer can then select the exposed methods and add them to the visual workflow. This is an extremely useful feature, since a major advantage of the M7 Application Assembly Suite is the ability to vary the access for each team member dependent on expertise. This prevents the novice J2EE programmer from causing serious problems.

To create an enterprise application, often developers must understand parts of all the layers involved from the connectors to the HTML, including data sources. Application programmers don't want to be J2EE experts. They need drag-and-drop centric solutions. They need uniform access to

components rather than the entangled accesses between layers that often afflict J2EE applications.


Often, because of lack of experience and schedule pressure, projects must settle for a subset of the J2EE architecture, using only the Web container and ignoring the other features. This is similar to purchasing a new automobile just so you can use the trunk!

There are some good visual JSP builders on the market, but the M7 Application Assembly Suite's JSP builder definitely is the best I've used (see Figure 2). WYSIWYG application builders are plentiful, but few if any support the full J2EE architecture. The M7 Application Assembly Suite comes closer than any I have seen.

While very impressive, however, the M7 Application Assembly Suite is not perfect. There is no easy way to use message-driven beans in the current workflow paradigm. Also, it has not yet been fully tested on WebSphere Application Server 5.0. Like many software development tool suites, I think the biggest area for improvement is in integration with other environments, such as IDEs and frameworks like Struts.

M7 is addressing these problems. Changes slated for this summer include repository integrations with JBuilder and Eclipse so developers and architects can place components into the repository from the IDE as well as launch the IDE from the repository to access the code behind the entries. In addition, there will be a Struts integration that will let programmers import the struts-config file and manage applications built using Struts as the workflow layer.

### Conclusion

The M7 Application Assembly Suite allows you to build your application's workflow, but when you add the repository for component reuse and the M7 Studio for visual JSP editing, you really have a platform for visually building MVC J2EE and deploying applications. 

WebSphere  
DEVELOPER'S JOURNAL

## Advertiser Index...

COMPANY	URL	PHONE	PG
CANDLE	WWW.CANDLE.COM	800-898-4426	26-27
DIRIG SOFTWARE	WWW.DIRIG.COM	603-889-2777	19
GLOBAL KNOWLEDGE	WWW.GLOBALKNOWLEDGE.COM	800-COURSES	25
JAVAONE	WWW.JAVA.SUN.COM/JAVAONE/SF	650-372-7054	31
KENETIKS	WWW.KENETIKS.COM	888-KENETIKS	13
LINUX BUSINESS & TECHNOLOGY	WWW.SYS-CON.COM/LINUX	201-802-3020	49
MACROMEDIA	WWW.MACROMEDIA.COM/GO/WSDJ/	415-252-2000	51
MOS SOFTWARE	WWW.MOS SOFTWARE.COM	952-345-8720	6, 7
PRECISE	WWW.PRECISE.COM/WSDJ	800-310-4777	5
PROLIFICS	WWW.PROLIFICS.COM/WEBSERVICES	800-675-5419	2
QUEST SOFTWARE	HTTP://JAVA.QUEST.COM/QCJ/WDJ	800-663-4723	15
TRILOG GROUP	WWW.FLOWBUILDER.COM	800-818-2199	52
VERSATA	WWW.VERSATA.COM/BUSINESSLOGICDESIGNER	800-984-7638	11
WEB SERVICES EDGE WEST 2003	WWW.SYS-CON.COM/WEBSERVICES2003WEST	201-802-3069	37
WILY TECHNOLOGY	WWW.WILYTECH.COM	888-GET-WILY	3

WebSphere  
DEVELOPER'S JOURNAL

## Coming Next Month...

### INTERVIEW

#### e-business on demand:

*A conversation with IBM's Kerrie Holley*

BY JACK MARTIN

### DEFECT RESOLUTION:

#### Closed-Loop Change Management:

*Addressing the complexity of coordinating complex projects*

BY PATRICK MERRITT

### WEB SERVICES:

#### Web Services Invocation Framework, Part 2:

*WSIF with different providers, and service implementation using WSIF*

BY BORIS LUBLINSKY

### SECURITY:

#### Understanding Tivoli Access Manager for WebSphere Application Server:

*Centralized management of EJBRole security*

BY EDWARD MCCARTHY



# Web Services Invocation Framework

## Part 1: WSIF architecture

— BY BORIS LUBLINSKY —



### 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 service-oriented architecture for the company. 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 technical architecture.

### E-MAIL

boris.lublinsky@cna.com

Today's most popular Web services APIs – JAX-RPC and JAXM – support two very different programming models for invocation of Web services, one synchronous, one asynchronous. If users need both models in a single application, they are forced to use two sets of very different APIs. This article, the first of a two-part series, describes an architecture and programming model – the Web Services Invocation Framework (WSIF) – that provides a single set of APIs that supports both models.

**J**AX-RPC, which is currently part of J2EE and consequently is a mandatory implementation for all J2EE application services vendors, defines and uses an XML-based remote procedure call mechanism. JAX-RPC is a very powerful, easy-to-use API for RPC-style Web services communications. The relative simplicity of the RPC model, which is very similar to the method invocation of the local Java class, explains its widespread popularity. Nearly all Web services tool vendors provide out-of-the-box support for these APIs, allowing for automatic generation of JAX-RPC stub- and proxy-based service implementation (either Java class or EJB). Incorporation of the decorating filters pattern allows for a very powerful mechanism for extending the JAX-RPC programming model and support for additional emerging Web services standards.

The drawback of using JAX-RPC, according to some experts, is that it usually leads to the creation of brittle applications. Remote procedures are strongly typed and possess a static signature that defines fixed parameters and return values. If there are any changes in the procedure signature, all dependent clients must be modified to reflect these changes. In addition, RPC by definition is a synchronous programming model, so a

caller cannot continue execution until the procedure has returned. Although asynchronous invocations are occasionally supported or can be simulated, most JAX-RPC procedures are synchronous, and therefore not responsive to changing network conditions. This leads many experts to believe that RPC-style invocations are not applicable to building evolving, fault-tolerant or fault-recoverable systems.

The Java API for XML Messaging (JAXM) was introduced to assist developers in creating business-to-business messaging applications using XML. JAXM defines an asynchronous, loosely coupled, document-based messaging model, which is at the moment less used, but allows achieving the full value offered by Web services. JAXM was designed to support the SOAP (and SOAP with attachments) messaging infrastructure over different transports, including, but not limited to, HTTP, MOM (most notably JMS), and mail.

JAXM provides an implementation of very powerful messaging APIs, allowing for the creation of an extremely flexible and extensible messaging infrastructure. The fact that JAXM does not dictate the content of the XML messages (only SOAP as an enveloping mechanism) allows system designers to use a very wide variety of messaging approaches, ranging from strongly typed messages using XML Schema definitions or JAX-RPC data type mappings to very complex semantic messages that provide very loose coupling between message senders and receivers.

The JAXM specification and reference implementation provide good support for synchronous messaging but are inadequate for implementing asynchronous messaging, experts agree. It is impossible to send an asynchronous message using JAXM without an explicit destination and a specific profile. (A profile is a domain-specific application of SOAP such as ebXML.) Therefore, usage of a particular profile requires specific knowledge of the implementation-specific profile classes involved, as there are no generic interfaces supported by all profile implementations. The requirement to use a specific profile to send an asynchronous message through a provider renders it impossible to send an arbitrary

document-based SOAP message to an arbitrary destination using asynchronous guaranteed message delivery.

Unlike JAX-RPC and JAXM, the Web Service Invocation Framework (WSIF) is an API that directly supports constructs of WSDL as an abstract service definition model. With WSIF the user programs directly to the abstract service representations expressed in WSDL, thus taking full advantage of the flexibility and richness of this service definition model, which:

- Views network applications as encapsulated services identified by their ability to exchange messages according to a set of predefined patterns – their supported portTypes
- Provides a standard-, protocol-, and platform-independent, unified view of network applications
- Separates core functionality from protocol and deployment details
- Allows clients to optimize access to desired functionality based on available protocols

WSIF allows for support of both synchronous and asynchronous communications. It also allows either usage of a fixed set of parameters and return values (similar to RPC-style communication) or sending arbitrary XML documents (wrapped as SOAP documents) as service requests and replies, similar to JAXM.

WSIF APIs also provide binding-independent access to any Web service. WSIF allows completely dynamic usage of different transport and transport support implementation for invocation of Web services based on examination of the metadata about the service at runtime. It also allows updated implementations of a binding to be plugged into WSIF at runtime.

I will describe the WSIF architecture and programming model. I will also provide code examples and explain how it works.

## WSIF Architecture

WSIF is composed of three major parts:

- **WSIF kernel:** Implements Java APIs that allow clients to program against an abstract model of the service, without any concern about the specifics of communication transport or message packaging structure.
- **WSIF providers:** Implement WSIF support over different encoding and transport implementation. WSIF is shipped with providers for Java, EJB, native JMS, Apache SOAP (over HTTP and JMS), Apache Axis (over HTTP and JMS), SOAP over RMI, and J2C (Java Connectors).
- **WSIF utilities:** Implement support functionality for both the WSIF kernel and WSIF provider implementations.

The architecture of WSIF is based on a number of factories. The reason for using factories is that they allow for hid-

ing the provider's implementation, providing for the use of the same interface, regardless of the actual provider used.

WSIF implementation encompasses the following main classes, defined in the `org.apache.wsif` package:

- **Service Factory class:** The `WSIFServiceFactory` class is a root class in WSIF hierarchy that allows for the instantiation of a service from a WSDL document at a known URL or known location on the hard drive. Service instantiation is done by parsing the WSDL document and building an internal service model.
- **Service class:** A `WSIFService` is a factory via which `WSIFPorts` are retrieved. A specific port can be requested by using its name, or a service can choose a port on its own, either based on the internal algorithm or port preference, which can be set by a service class. An actual port implementation is part of the provider implementation. The WSIF environment stores a list of providers that can each support a particular WSDL binding. The service looks up providers based on the port binding style, for example, SOAP binding, JMS binding, Java binding, EJB binding, etc. If there is more than one provider supporting a given binding, the `WSIFService` implementation can choose a provider based on a number of criteria, for example, the preferred provider for a particular binding, the first defined provider, etc.
- **Port class:** `WSIFPort` implements a factory method for `WSIFOperations`. A `WSIFPort` handles the details of creation of an operation based on the port definition.
- **Operation class:** The `WSIFOperation` is the runtime representation of an operation. It is responsible for invoking a service based on a particular binding. It provides methods to create input, output, context, and fault messages, and to invoke the operation.
- **ResponseHandler class:** The `WSIFResponseHandler` is an implementation of the `Callback` object, used for asynchronous operation invocation using the `executeAsyncResponse()` method on the `WSIFOperation` class.

WSDL	WSIF
UDDI	<code>WSIFServiceFactory</code> or <code>JNDI</code>
Service	<code>WSIFService</code>
Port	<code>WSIFPort</code>
Binding	No corresponding entity
Operation	<code>WSIFOperation</code>
Message	<code>WSIFMessage</code>
Part	No corresponding entity

**TABLE 1: CORRESPONDENCE BETWEEN WSDL AND WSIF ARCHITECTURES**



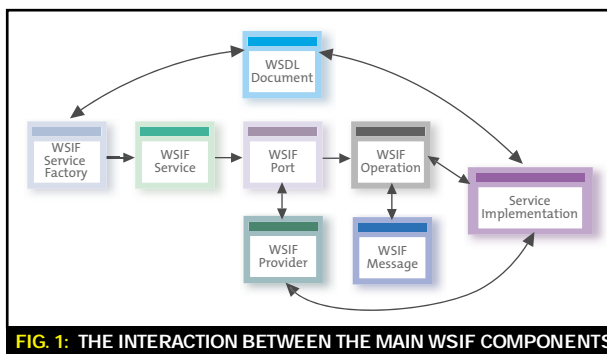
- **Message class:** WSIFMessage implements WSDL messages, which are composed of named parts tied to some type system, normally an XML Schema. To invoke such services, this schema type is mapped to possibly more sophisticated type systems, provided by the programming language in use. For instance, the schema type is mapped to certain Java classes by serialization and deserialization for cases where Java is the native type system. The design of WSIFMessage allows you to employ any native type system as a message part. Parts in a message can be typed by employing different type systems. This is needed if parts from multiple separate WSDL messages are associated with different native type systems and have to be combined into a single message.
- **Provider class:** A WSIFProvider is an implementation of a WSDL binding. It can run a WSDL operation over a protocol that depends on a specific binding. WSIF providers implement the bridge between the WSIF API and the actual implementation of a service. WSIF providers are currently available for SOAP over HTTP, SOAP over JMS, Java, EJBs, and native JMS. New providers can easily be added to the WSIF framework.
  - The SOAP provider allows WSIF stubs and dynamic clients to invoke SOAP services.
  - JMS is a messaging technology. The mapping to a JMS destination is defined during deployment and maintained by the container.
  - The WSIF Java provider allows WSIF to invoke Java classes and JavaBeans; local Java code running on a JVM can be incorporated easily.
  - The EJB provider allows WSIF clients to invoke Enterprise JavaBeans. The EJB client JAR must be available in the client runtime with the current provider.

Table 1 shows the correspondence between WSDL and WSIF components. Figure 1 illustrates the interaction between the main WSIF components.

There are two ways of using WSIF: as a stub-based model and as a dynamic invocation interface (DII). The stub-based approach allows you to use any JAX-RPC compliant tool (for example WSDL2Java) to generate stub interfaces corresponding to WSDL portTypes. It allows usage of WSIF, pretty much the same as JAX-RPC with multiple provider protocol support. In my evaluation I concentrated on the DII approach to WSIF invocations.

## WSIF Programming

I have been testing WSIF with WebSphere Studio Application Developer (WSAD) 5, which allowed me to generate some of the code used and test the code from



within WSAD itself. Evaluation involved the following steps:

- Creation of the basic service implementation and WSDL, defining the service
- Creation of the service client, allowing for the invocation of the service using SOAP/HTTP
- Using the same client to use other providers, including:
  - An EJB provider
  - A native JMS provider (WebSphere MQ as JMS provider)
  - SOAP over JMS provider (WebSphere MQ as JMS provider)
- Using JNDI for storing the service information
- Creating custom SOAP headers
- Using WSIF to send custom XML files
- Asynchronous service invocation using WSIF

## Creation of the Basic Service Implementation

I've used the AddressBook EJB provided by the WSIF distribution as a sample of the basic Web service implementation. The code for this bean can be found in the subdirectory samples/ejb/service of the WSIF distribution. Additional classes defining types used are WSIFAddress and WSIFPhone, located in the subdirectory samples/ejb/service/addressbook/wsiftypes. After compiling and building the code within WSAD, I used WSAD wizards to generate the WSDL files for this service. This is a standard WSDL file, presented in Listing 1 (All of the code for this article can be downloaded from [www.sys-con.com/websphere/sourcecfm](http://www.sys-con.com/websphere/sourcecfm)). The most basic WSIF client implementation is presented in Listing 2.

The basic WSIF client is composed of the following steps:

1. First we need to get a new instance of WSIFServiceFactory.
2. WSIFServiceFactory is then used to create a WSIFService, which is built based on the WSDL file. The location of the WSDL file can be specified either as a URL location available over HTTP or as a file-based location available through the classLoader. In this simple example I use a URL address. Additional parameters for service creation are service name/namespace and port name/namespace. Because of the simplicity of our WSDL file, containing only a single service with a single port, we will set these values to null. When WSIFService is created, a model of the service (based on the WSDL) is built internally. After the service is created, type maps must be set for it if we want to set message parts using Java objects. Setting type maps creates a correspondence between the Java class types and the namespace/names used in the WSDL for the message definitions.
3. WSIFService is used for creation of WSIFPort. The port implementation is selected based on the binding type and providers available. A list of the available providers' names is contained in the file org.apache.wsif.spi.WSIFProvider, which is part of the WSIF.jar and is located in the META-INF/services package. When the port is created for the first time this file is loaded into memory and the list of providers is used every time a port is created. Each provider can be queried about the binding and protocol types that it supports. If these two correspond to the binding and protocol types specified in the WSDL for the service/port pair, the provider can be used for the service invocation. There are three situations that can occur when choosing a provider for service invocation:



- A preferred provider can be set for a particular binding type using this command; here we are specifying usage of the AXIS provider for the SOAP binding:

```
WSIFPluggableProviders.overrideDefault
Provider(
```


```
"http://schemas.xmlsoap.org/wsdl/soap/",
new WSIFDynamicProvider_ApacheAxis());
```

- If there is only one provider available for a particular binding/transport type, this provider is used.
  - In the case where multiple providers are available for the required binding/transport type, the providers will alternate in round-robin fashion.
4. The next step is creation of the WSIFOperation using WSIF-Port. An operation is created using a service name or a combination of service name and input and output messages.
  5. WSIFOperation is then used to create input, output, and fault messages.
  6. The input message now has to be populated. Population of the input message is done by setting parts (defined in the WSDL) of the input message. It is possible to set message types using Java classes based on the type map, which is set by the service.
  7. The last step is a service invocation. Invocation returns a Boolean variable that is equal to true if the operation succeeds, or false otherwise. In the case of failure, a fault message can be examined to determine the exact reason for failure. In this case a fault message is populated based on the SOAPBody: Fault element defined in the SOAP specifi-

cation. In the case of successful service invocation the content of the output message can be extracted into Java classes based on the output message parts (similar to the population of the input messages). Following is the result of execution of the simple WSIF client:

```
WSIF factory created
WSIF service created
WSIF types map created
WSIF port created
WSIF operation created
WSIF input message created
Successfully added name and address
Successful lookup of name 'Jack' in
addressbook
The address found was:
streetNum=1
streetName=The Waterfront
city=Some City
state=NY
zip=47907
phoneNumber=areaCode=765
exchange=494
number=4900
```

### Conclusion

WSIF offers many advantages over other Web services invocation frameworks. In Part 2 I will discuss more advanced topics of WSIF programming, such as JNDI bindings, using different providers, asynchronous service invocation, and messaging. 

www.sys-con.com



Millions  
of Linux Users  
One Magazine

## Linux Business & Technology

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. *Linux Business & Technology* is aimed squarely at providing this group with the knowledge and background that will allow them 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. *Linux Business & Technology* will not feature low-level code snippets but will focus 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 will see a different focus, allowing a detailed analysis of all the components that make up the greater Linux landscape.

### Regular features will include:

Advice on Linux Infrastructure  
Detailed Software Reviews  
Migration Advice  
Hardware Advice  
CEO Guest Editorials  
Recruiting/Certification Advice  
Latest News That Matters  
Case Studies

 SYS-CON  
MEDIA

The World's Leading Technology Publisher

SAVE 30%  
OFF!

REGULAR ANNUAL COVER PRICE \$71.76  
YOU PAY ONLY  
\$49<sup>99</sup>

12 ISSUES/YR

\*OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**LINUXEDGE**  
conference & expo

June 3-5.....LONDON  
June 24-26.....BERLIN  
September.....HONG KONG  
October.....CALIFORNIA

SUBSCRIBE  
TODAY!

WWW.SYS-CON.COM  
OR CALL  
1-888-303-5282

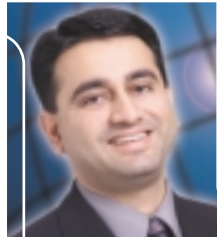
FOR ADVERTISING INFORMATION:  
CALL 201 802 3020 OR  
VISIT WWW.SYS-CON.COM

ALL BRAND AND PRODUCT NAMES USED ON THIS PAGE ARE TRADE NAMES, SERVICE MARKS, OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.

# Improving the Decision-Making Process

BY SUMIT DESHPANDE

Information that is readily available to the right people at the right time, and that enables them to act quickly and decisively, is a crucial requirement of enterprises that intend to grow and remain profitable. The intelligent delivery of information must obtain maturity levels consistent with enterprise goals and must employ the appropriate technology to facilitate this purpose. In today's enterprise environment, users must consolidate data and deliver it as useful information and knowledge that empowers the right business decisions.



information access, relevancy, and analysis. The model defines four levels of information delivery. As organizations progress through the levels, they attain increasingly detailed information access and analysis to improve and automate the business decision-making process. Each level targets specific information delivery challenges, and once users achieve a certain level, they encounter new challenges that encourage them to aspire to the next level. Users also realize that it is virtually impossible to go back to lower levels, simply because they cannot afford to compromise efficiencies already achieved.

Information delivery maturity can be described as the ability of an enterprise to provide the correct information, to the correct person or system, at the correct time, in the correct format, in an intuitive, flexible, and secure manner, so that the correct action can be taken. Information delivery requirements include making data centrally accessible and available, processing the data into trustworthy information that is understood by the user, presenting information in the context of the user's role, and using the information to automate appropriate processes. There are four levels of information delivery maturity, and as enterprises grow from one level to the next, they gain the ability to control and use information within the organization.

Business and IT executives are under increasing pressure to improve effectiveness and efficiency while cutting costs, creating value from IT investments, and demonstrating ROI. The key is information that, unfortunately, is often difficult to find. A December 2002 Integrated Solutions study found that business professionals spend up to 50% of their time looking for pertinent information, make an average of 19 copies of each document, spend an average of \$120 in labor searching for each misfiled document, lose 1 out of every 20 documents, and spend an average of 25 hours re-creating each lost document. Other barriers to timely information delivery include:


- The data exists in heterogeneous formats and in different locations.
- Just because data is available doesn't mean users have the information or knowledge they are seeking.
- Depending on the structure of an enterprise, information sharing may be problematic.
- Information may be available, but not necessarily relevant.
- Application of intelligence to information in order to facilitate automatic action is a complex process.

The Information Delivery Maturity Model provides a basis for addressing the challenges and opportunities of knowledge delivery within an organization, and identifying key requirements for improving

information access, relevancy, and analysis. The model defines four levels of information delivery. As organizations progress through the levels, they attain increasingly detailed information access and analysis to improve and automate the business decision-making process. Each level targets specific information delivery challenges, and once users achieve a certain level, they encounter new challenges that encourage them to aspire to the next level. Users also realize that it is virtually impossible to go back to lower levels, simply because they cannot afford to compromise efficiencies already achieved.

- **Level 1:** Focuses on centralizing access to data. Users save time because they know where to go in order to get the data they need. Data on paper and other physical media needs to be digitized for easy access and longevity. Manual processes are automated to the furthest extent possible, and people are given appropriate access to the data they require to do their jobs.
- **Level 2:** Focuses on transforming raw data into information that is objective, trustworthy, and usable. This often involves automatic refining, sorting, and analysis of data to present information to the user in a manner that is clearly understood.
- **Level 3:** Involves the application of operational intelligence tools to extract and deliver relevant information to decision makers, taking their individual roles into consideration. Filtering out the unnecessary and delivering only relevant knowledge saves significant time and enables users to make better decisions quickly.
- **Level 4:** Empowers users to set up custom rules for their unique roles, responsibilities, and experiences so as to receive the appropriate information and automate the necessary response.

Despite long-time availability of systems – including knowledge management, business intelligence, decision-support, and executive information systems – many organizations have yet to attain intelligent information delivery. Others are struggling to move beyond simple data access into more sophisticated analysis and personalization. Each level of information delivery maturity provides great value to an organization, depending on the needs of the business at a given time.

Managing information and delivering knowledge to decision makers within the enterprise is complex, but by no means impossible. With an understanding of an enterprise's information delivery maturity, users can leverage information assets to provide maximum business value. 

**ABOUT THE AUTHOR...** Sumit Deshpande is a technology strategist in the Office of the CTO at Computer Associates. He defines and communicates CA's global strategy for wireless technology and business intelligence solutions. Sumit has a broad range of IT experience, including networking, application development, technology consulting, and market analysis.

E-MAIL

sumit.deshpande@ca.com

# MACROMEDIA

[WWW.MACROMEDIA.COM/GO/WSDJ/](http://WWW.MACROMEDIA.COM/GO/WSDJ/)

# TRILOG GROUP

[WWW.FLOWBUILDER.COM](http://WWW.FLOWBUILDER.COM)