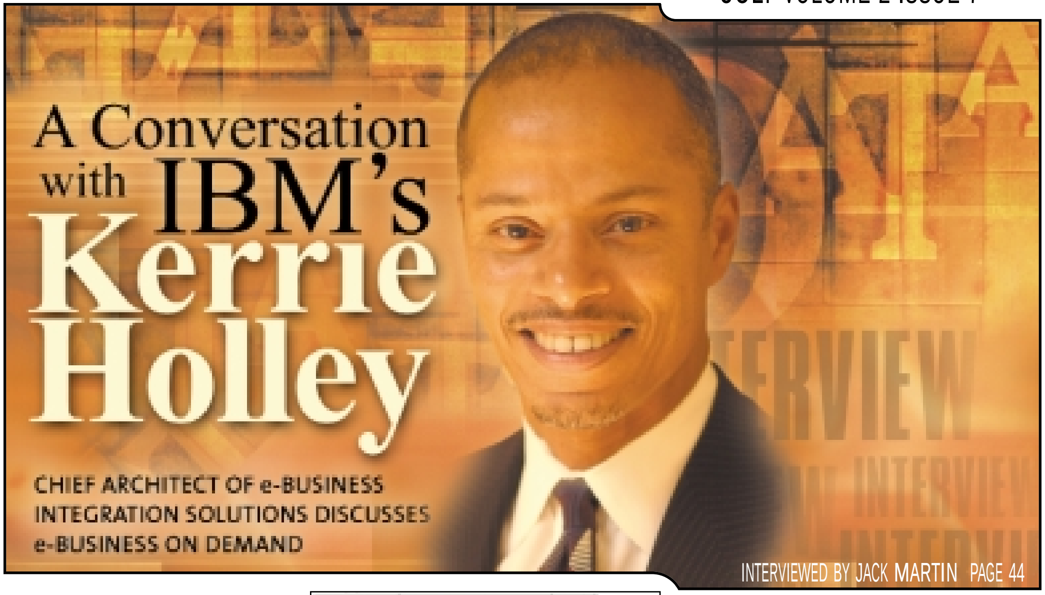
ON-DEMAND COMPUTING: THE NEXT BIG THING?

WebSphere® Developer's Journal

The World's Leading Independent WebSphere Developer Resource

WebSphereDevelopersJournal.com

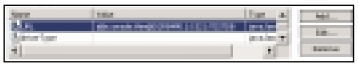
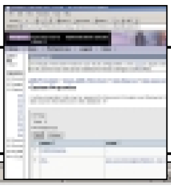
JULY VOLUME 2 ISSUE 7

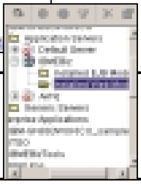


A Conversation with IBM's Kerrie Holley

CHIEF ARCHITECT OF e-BUSINESS INTEGRATION SOLUTIONS DISCUSSES e-BUSINESS ON DEMAND

INTERVIEWED BY JACK MARTIN PAGE 44



DISPLAY UNTIL SEPTEMBER 30, 2003
$8.99US $9.99CAN
07
0 09281 03422 3
SYS-CON MEDIA

PROLIFICS

WWW.PROLIFICS.COM/WEBSERVICES

WILY TECHNOLOGY

WWW.WILYTECH.COM

Software Needs to Create Value

Focus on solving real-world problems

BY JACK MARTIN

Another month has passed, and I have been hearing from an awful lot of readers lately on the topics of deflation and the general condition of the economy as it relates to enterprise software.

To answer their questions requires that we take a quick survey of the economy. The reason we do this is the same reason any company ever buys software – to assist in its primary business. So off we go to see how the people who are supposed to buy software are doing in their own businesses.

Consumer spending remains sluggish, as people are just not buying things the way they were a year ago – and even that was not a good time for retailers. Manufacturing is mixed at best; some areas report signs of improvement, while others are still seeing declines in orders. The service sector is still experiencing a plodding order velocity.

Although low mortgage rates continue to stimulate residential construction and home sales in most districts, commercial construction and real estate markets are still very weak. You can verify this on your way to work. You'll see a whole lot of For Rent and For Sale signs if you work in a central business district or an area that has a concentration of office buildings.

The energy industry continues to strengthen. I'm sure you have been paying more for gas and to heat your home. Agricultural production has been drenched by wet weather in some areas. If you live in the Northeast, it's rained so much that you are probably a little soggy as you read this editorial. Most areas are still experiencing layoffs and lack of new job creation, which of course creates downward pressure on wages, although benefit costs continue to rise.

What this means to you is that business is not good unless you're a mortgage broker, real estate agent, or home builder. The real estate market will go down the hopper too if we experience even mild deflation. After all, who wants to own a house that is worth progressively less and less – eventually becoming worth less than the mortgage amount. When home values drop below what is owed, lots of people walk away from the payments and the homes, putting major downward pressure on housing prices.

The single best way to get some sales going in software is to solve some of the real-world problems all businesses are currently encountering. Unfortunately, the bulk of enterprise-class software is designed to accommodate ever-expanding businesses in an ever-expanding economy.

Currently, the big play is to integrate all kinds of back ends into a cohesive whole from the user's perspective. Business integration does not allow the company that has purchased the software and services to integrate everything with one new sale; what it does is achieve a cost saving. The reason this cost saving is possible is that the initial work was poorly done by the vendor and the business's in-house staff.

The reality is that with the exceptions of manufacturing design and graphic arts, all business software works in some kind of a score-keeping role like accounting, inventory, order systems, POS, CRM, and ERP. It does nothing more, nothing less.

Software currently creates nothing. The time has come for someone to introduce the first software product that actually creates true value.

What the business software world needs now is a software product that increases the sales that can be counted at the end of each business day. Increasing sales means products or services have been delivered to a willing buyer for cash. If you are reading these words and an idea has come to you, go for it. The world will beat a path to your door when you are finished.

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

WebSphere® DEVELOPER'S JOURNAL

ADVISORY BOARD

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

EDITOR-IN-CHIEF	JACK MARTIN
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 STILLEY

WRITERS IN THIS ISSUE

Roger Berger, Richard Ferri, Troy Holmes, Moon J. Kim, Adam Kolawa, Boris Lublinsky, Jack Martin, Dikran Meliksetian, Patrick Merritt, Geetha Ramaswamy, Sankar Venkitachalam

SUBSCRIPTIONS

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

PRESIDENT AND CEO	FUAT 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	NIKI PANAGOPOULOS SHELIA DICKERSON
CUSTOMER RELATIONS MANAGER	RACHEL MCGOURAN

EDITORIAL OFFICES

SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
WebSphere® 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

WORLDWIDE NEWSSTAND DISTRIBUTION
CURTIS CIRCULATION COMPANY, NEW MILFORD, NJ

GLOBAL KNOWLEDGE

WWW.GLOBALKNOWLEDGE.COM

MQSOFTWARE

WWW.MQSOFTWARE.COM

MQSOFTWARE

WWW.MQSOFTWARE.COM

Grid Computing

Grids will lead to a new world of on-demand computing

– BY RICHARD FERRI, MOON J. KIM, AND DIKRAN MELIKSETIAN –

In October 2002, when IBM CEO Sam Palmisano ushered in the new age of on-demand computing to a group of 300 customers, some asked whether the on-demand initiative was just more marketing hype – or a fundamental change in the way customers will view computing in the future. The answer is that on-demand computing represents a fundamental change in how customers will use technology.

ABOUT THE AUTHOR
Richard Ferri is a senior programmer in IBM's Linux Technology Center, where he writes about and works on open-source cluster projects. He recently joined the Advanced Linux Response Team, working with customers converting to Linux. He holds five patents in the field of clustering, with one pending.

E-MAIL
rcferri@us.ibm.com

To understand this change requires a thorough understanding of grid computing and how it complements IBM's on-demand strategy. We will explore the nature of grids, the difference between clusters and grids, and how dynamic virtual organizations can use grids to share information and solve problems in an on-demand world. We will explain why protocols are important to grids, and conclude with a snapshot of perhaps the best-known grid, the TeraGrid.

Specifics of On Demand

Systems for an e-business on demand should be empowered by responsive, flexible, focused, and resilient technologies such as autonomic computing technologies. An on-demand autonomic computing approach enables a network of organized, self-managing computing components that provides customers with what they need, when they need it, with less effort.

The dynamics of the market are changing, and that requires a new way of thinking about business processes and the information technology infrastructure that supports them. IBM is entering the next phase of e-business, in which companies move beyond simply integrating their various processes, to a world in which they need to be able to sense and respond to fluctuating market conditions in real time. IBM is leading the way toward the on-demand world, and has the business insight and technological expertise to help customers become on-demand businesses.

An on-demand business is an enterprise whose business processes – integrated end to end across the company and with key partners, suppliers, and customers – can respond with agility and speed to any customer demand, market opportunity, or external threat. An on-demand system is dynamically responsive to business environment changes. It uses variable structures and adapts with flexibility to business environment changes. This flexibility will enable it to reduce risk and to perform at high levels of productivity, cost control, capital efficiency, and financial predictability. It is focused on its core competencies and differentiating tasks and assets. Finally, it is resilient enough to manage changes and threats.

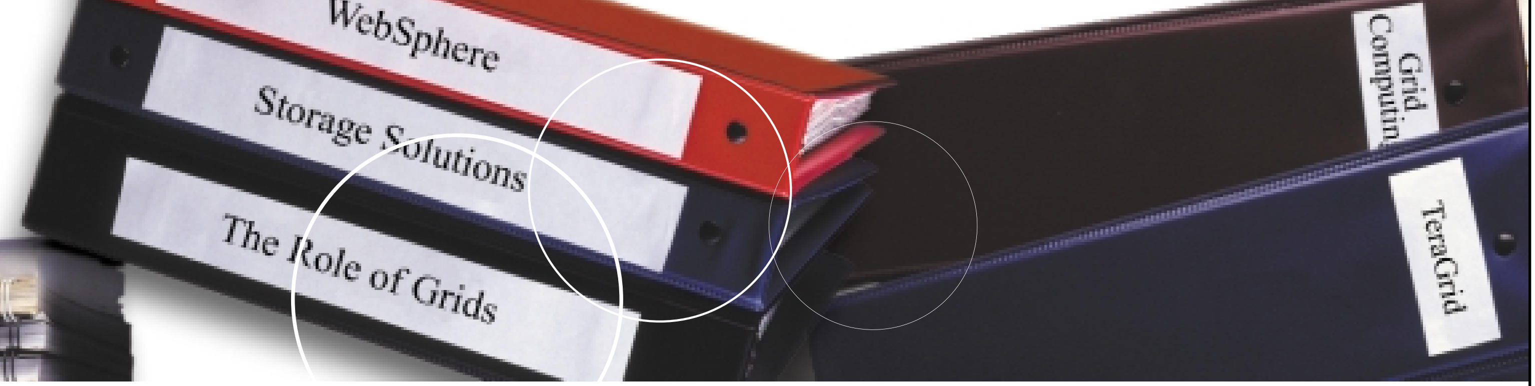
In an on-demand business the computing environment is integrated to allow systems to be seamlessly linked across the enterprise and its entire range of customers, partners, and suppliers. It uses open standards, so different systems can work together and link with devices and applications across organizational and geographic boundaries. It is virtualized to hide the physical resources from the application level and make the best use of technology resources and minimize complexity for users. The grid computing model – which makes the collective power of the computing resources in the grid available to any user – is a good example of a virtualized system model.

The On-Demand Operating Environments

To be successful in the future, businesses need to achieve new levels of productivity by becoming fully integrated – by integrating end to end across people, processes, and information. To accomplish this goal, the business must build on an IT infrastructure specifically designed for that purpose. This infrastructure is called the on-demand operating environment. Specifically, the on-demand operating environment depends on integrating middleware, open standards (so the company can interact with anyone in the world), virtualized systems (to hide the complexity of the infrastructure and use its resources more efficiently), and autonomic capabilities (to manage the growing system complexity).

Software On Demand

As part of an on-demand operating environment, software on demand is designed to address the heterogeneity and



complexity of today's systems so that the existing computing power is fully utilized, and data can be shared among partners, customers, and suppliers responsively. Virtualization technology, incorporated into WebSphere, makes it easier for a business to integrate processes and IT solutions that have been traditionally separate. Using WebSphere in a grid environment enables the business to manage applications running on different servers – with differing priorities, usage patterns, and computing profiles – as a single environment.

Storage Solution

In an on-demand environment, with quickly changing requirements for data sharing, businesses can no longer afford the simplistic thinking of a storage device being "owned" by a server. Storage virtualization, the layer of software that isolates the physical storage from the application servers that use them, provides a centrally managed method of sharing storage within an environment. In a virtualized environment, physical disks are pooled and can be mapped into virtual disks and used by all servers. Virtualization also removes the complexity of individually managed physical disks and allows customers to maximize usage of their storage area network.

These new, specific requirements of on-demand operating environments force us to think differently about the very nature of computing. The barriers that inhibit sharing of information and resources within an entity or between entities have to be carefully removed to permit dynamic sharing of resources and information by disparate groups, for disparate lengths of time. This dynamic sharing of resources is exactly what grids are designed to do.

The Role of Grids

Clusters, by definition, are interconnected whole computers that cooperate to solve a problem. But behind the definition, clusters are fairly static and homogeneous creations. The number of nodes in a cluster might grow or shrink over time, but the intent of the cluster rarely changes (and if it does, it's usually with a significant amount of re-architecting). For example, a cluster assembled for purposes of weather forecasting or for geopetrol exploration will have software installed for those applications only. The individual users of the cluster may change over time, but typically even as users turn over, the applications they run on the cluster remain similar to those run by previous users. The types of data that cluster users manipulate are typically chosen during the creation of the cluster.

Second, clusters are usually homogeneous, both in function and hardware. With the possible exception of the cluster manager, the purpose of the nodes in a cluster is to solve the same kind of problem. Therefore, the nodes of a cluster are usually machines with similar architectures and operating systems. They also have, in general, a uniform set of software libraries and applications. Finally, for ease of administration and access, clusters are generally in one physical location.

However, grids can be thought of as clusters without the preconceived limitations. Grids, like clusters, are interconnected whole systems that cooperate in solving problems. However, a grid can be thought of as a cluster that forms dynamically in response to requirements based on computing power and data sharing (with its associated authorization issues), without regard for the physical location where the computing power is generated, where the data is coming from, or on what kind of platform it is available. Imagine, in essence, having the ability to carve the necessary computing resources out of the universe of computing resources available to solve a specific problem, for whatever duration is required. A grid allows the dynamic, on-demand assembly of the different types of computing resources necessary to solve a particular problem. These computing resources usually belong to separate organizations whose members are interested in the solution of a given problem. This is the thinking that fostered the grid concept. The premise of grids is to share computing resources and data in a dynamic way on an unprecedented scale.

Virtual Organizations and Grids

At the heart of understanding the grid paradigm is the concept of virtual organizations, or VOs. A VO is set of individuals and/or institutions that are dynamically brought together to solve a problem. The VO might span multiple companies, academic organizations, or government departments that have an interest in solving a common problem. Interactions among members of the VO are governed by a well-defined set of rules. The sharing of resources is highly controlled, "with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs," according to Foster, Kesselman, and Tuecke in their paper, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*.

For example, with grid computing, the design and manufacture of a new model of automobile can become an increasingly international and collaborative event. Car designers in Germany may want to provide detailed specification information to manufacturing plants in England. Multiple manufacturing sites in England may want to share inventory, supplier, and pricing information. The sales team in the United States may want to share real-time information on how the manufacture of a single automobile is going as an anxious customer awaits delivery of the vehicle he or she ordered. A grid converts the design, manufacturing, and sales teams – along with the customer – into a virtual organization.

Scientists working to solve a problem in medicine might also combine to form a long-term virtual organization. A team brought together to forge an emergency response in case of disaster, however, might bring together individuals

ABOUT THE AUTHOR
Dr. Moon J. Kim is an IBM senior technical staff member responsible for the development of strategic infrastructures. He has developed system solutions such as the Advanced Web System and the Broadband Interactive System, and was also a key architect and developer of the S/390 family and MPP systems. He is an IBM Master Inventor and has published numerous technical papers.

E-MAIL
moonkim@us.ibm.com

and institutions representing relief response agencies, medical response teams, and law enforcement. Such a dynamic team would be formed in real time in response to an emergency, and would last only for the duration of the relief effort.

Each of these VOs needs to collaborate in response to a problem. VO members are probably from different geographies, may work for different companies or entities, and will provide to and take from the grid different resources. VOs may be well thought out and planned prior to their creation, or they might be in response to a particular one-time, unforseeable event. Regardless of how a VO is formed, the need to share information and computing resources in real time requires individual systems within a grid to cooperate in well-defined ways by following various protocols.

The Role of Standards in Grids

The paradigms described in the previous sections would be very difficult to realize without established industry standards. If we want to make resources of various types belonging to different organizational entities collaborate in solving a common problem, it is necessary for these resources to communicate with each other using common standards.

The most significant activity in this area is the Open Grid Service Architecture (OGSA) definition. Drawing on prior research and experimentation in both the Web services arena and the grid computing field, OGSA defines an architecture for grid services. A grid service in this context is a Web service tailored to the requirements of a grid environment. This architecture abstracts and virtualizes the hardware and software platform on which the grid services will run. Thus, it provides a mechanism to allow grid nodes to interact with each other through grid services, irrespective of their hardware/software platforms. The core of the OGSA is the Open Grid Service Infrastructure (OGSI), which defines a set of interfaces that every grid service must expose and implement. OGSI is being standardized through the efforts of the Global Grid Forum (GGF), which is a community-initiated forum of researchers and practitioners working on distributed computing and grid technologies. GGF's primary objective is to promote and support the development, deployment, and implementation of grid technologies and applications via the creation and documentation of "best practices" – technical specifications, user experiences, and implementation guidelines.

The Globus Toolkit v3.0 (GT3) is an implementation of OGSA from the Globus Project. At the time of writing, the release of a production version of GT3 was planned for June 2003. The GT3 core is the reference implementation of OGSI. GT3 base services include the security service, and additional services available (see Figure 1) include a managed job service that allows remote execution of programs, a file transfer service, and an information service that exposes and indexes information about the other services and system information.

WebSphere

Over time, IBM products such as WebSphere will adopt the OGSA (see Figure 2) for middleware products. WebSphere will function as the Web services engine in the OGSA. OGSA will run on WebSphere and WebSphere with OGSA will be integrated into IBM servers and storage platforms.

TeraGrid

IBM is a key supplier to the TeraGrid, probably the most striking and best-known example of the grid principles in practice. The raw power of the TeraGrid is in itself impressive – 20 teraflops (a teraflop is a trillion floating point operations per second) of computing power, nearly a petabyte (a petabyte is just over a million gigabytes) of data, distributed over five U.S. sites and connected by a 40-gigabits/second backbone – the fastest research network in the world (see Figure 3). The five sites include the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign; the San Diego Supercomputer Center at the University of California, San Diego; Argonne National Laboratory in Argonne, IL; the Center for Advanced Computing Research at the California Institute of Technology in Pasadena; and the most recent addition, the Pittsburgh Supercomputer Center.

In true grid spirit, some of the sites specialize in computing power, and others specialize in data management and visualization resources, with each site contributing specific resources to the overall TeraGrid, comprising a whole greater than the individual parts. The peer-to-peer structure of the TeraGrid allows scientists to tap into the collective resources of the grid from their local workstations. Besides the general objective of enhancing collaboration, the TeraGrid project has several high-level objectives:

- To provide an unprecedented increase in the computational capabilities available to the open research community
- To deploy a distributed system, without centralized control, which allows resource mapping by participants
- To create an "enabling cyberinfrastructure" for scientific research so that additional resources can be readily added, and to provide a model for future grids

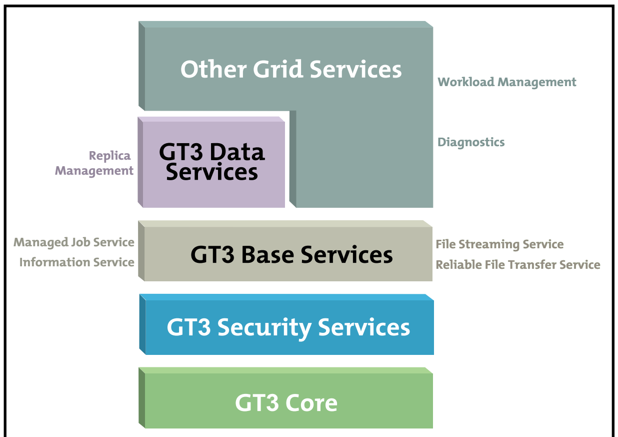


FIG. 1: GT3 ARCHITECTURE

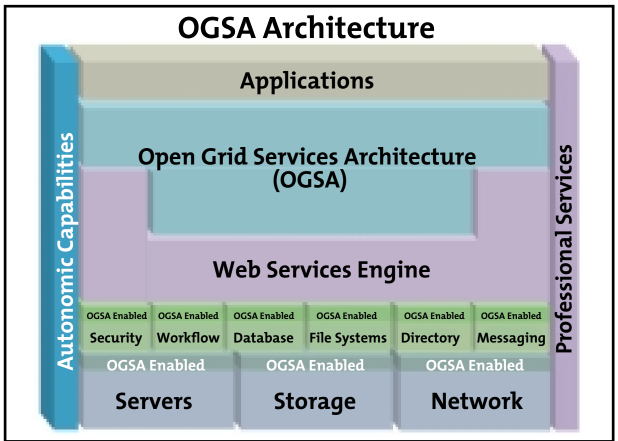


FIG. 2: OGSA ARCHITECTURE

ABOUT THE AUTHOR
Dr. Dikran S. Meliksetian works with the Internet Technology team at IBM and is involved in the design and development of advanced content management applications. He is a senior technical staff member and is engaged in the design and development of the IBM internal grid based on industry standards.

E-MAIL
dikran_meliksetian@us.ibm.com

VERITAS

WWW.VERITAS.COM

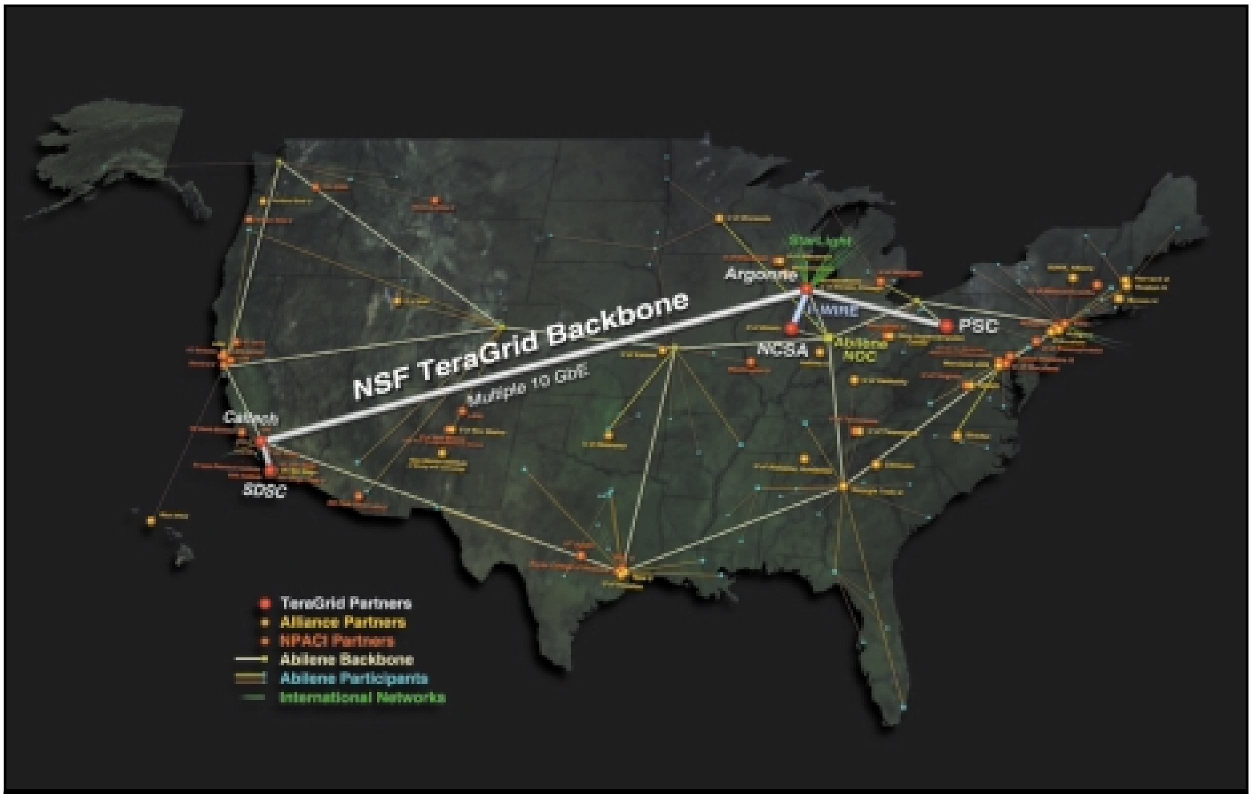


FIG. 3: THE TERAGRID

In practice, the individual members of the TeraGrid follow a set of standards that dictates a certain level of function, but not an implementation. This allows independent control of the individual sites, and the addition of new sites, provided they meet the functional requirements for participation. In practice, there is both a grid services layer and a set of application services. The grid services layer includes software that does overall grid scheduling, data movement, job scheduling and monitoring, and authentication and resource management. The application services are hosted on top of the basic and core grid services, and include support for batch job management, interactive job management, large-data management, and archival support.

The TeraGrid is the perfect match for the on-demand computing paradigm. It can be broken up into chunks to meet immediate computing needs, and the combined resources of the individual sites make it possible to run applications that otherwise would not be feasible, or that would require a physical presence at the computer. Since the TeraGrid has been designed to conform to the emerging grid standards (such as Globus, the de facto standard), and because the functional expectations for joining the grid have been architected, the TeraGrid is extensible. Other sites meeting the functional architecture will be added in the future. There is a feeling that the TeraGrid is the first major step toward a national cyberinfrastructure, a pervasive network of computers all running grid-standard software. Just like the power grid that the consumer simply plugs into without giving a thought as to where the power is generated, the cyberinfrastructure will provide transparent computing resources that over time may be adopted by the business sector. Already we're seeing advances in real-time weather forecasting and tornado prediction based on the on-demand power of the grid.

Conclusion

In this article, we've discussed the ways IBM has introduced the specifics of on-demand computing in the areas of operating environments, software, and storage. We've discussed the concept of virtual organizations and how they form and dissolve dynamically, with an ever-changing list of computing requirements that are difficult to plan for. We've discussed the concept of grids, how adherence to standards enables grids, and how grids differ from clusters in their ability to morph to meet the demands of virtual organizations. We've talked about the TeraGrid, which is providing scientists with an unprecedented ability to collaborate on a national scale, perhaps on the road to a national cyberinfrastructure. By its very nature, the grid is becoming the vehicle to lead us to a new world of on-demand computing.

References

- Foster I., Kesselman, C., and Tuecke, S. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations:* www.globus.org/research/papers/anatomy.pdf
- Catlett, C. *The TeraGrid: A Primer:* www.teragrid.org/about/TeraGrid-Primer-Sept-02.pdf
- Foster, I., Kesselman, C., Nick, J., and Tuecke, S. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration:* www.globus.org/research/papers/ogsa.pdf
- Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., and Kesselman, S. *Grid Service Specification (Globus Project, Draft 29)*: www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-29_2003-04-05.pdf
- Global Grid Forum: www.ggf.org

KENETIKS

WWW.KENETIKS.COM

Addressing the complexity of coordinating complex projects

Closed-Loop Change Management

BY PATRICK MERRITT

As software development projects become increasingly complex, coordinating the defect resolution process becomes critical.

ABOUT THE AUTHOR
Patrick Merritt is a product marketing manager for Merant, which delivers the industry's most flexible and comprehensive enterprise change management solutions. With more than eight years of experience at Merant, he has held a number of customer-facing roles and has been directly involved with technology implementations for customers throughout the United States.

E-MAIL
patrick.merritt@merant.com

"I thought you fixed that defect?"
"I thought you verified that I fixed the defect?"
"Wasn't this issue resolved?"
"Did you fix that defect yet?"
"Why isn't this defect closed; I fixed the problem last week!"

Sound familiar? Coordination among different people on a project team can be difficult. Simply coordinating the activities of multiple software engineers at one location may be easy, but try multiple locations and it becomes more difficult. Now mix in the quality assurance (QA) group – which has a different aspect of project responsibility – with the software engineers and you've increased the overall project coordination complexity.

Ultimately, each "silo group" within the project team has the same goal: to deliver a quality software product on time and within budget. Many engineering teams use software and processes to help solve this coordination problem, but how effective are these solutions today? Do they enable coordination among all team members? Do they ensure that issues are resolved and that the appropriate individuals are notified? Are they simple to use?

The defect resolution process is an everyday task during the development cycle of a typical software application – either an existing application or a new one being created. Figure 1 illustrates a simple workflow that shows the high-level steps in the defect resolution process.

For this typical scenario the test engineer would identify and document the defect, submitting a defect report or ticket. The defect would be assigned to the software engineer who would fix the defect. The test engineer would then verify that the defect is resolved and close the ticket.

Many organizations employ a defect-tracking solution in which defects are recorded by submitting defect reports and then tracked and managed throughout the defect resolution process. These systems, when used properly, can provide a closed-loop change management process. Closed-loop change management is simply a process ensuring that all identified issues are tracked, resolved, and communicated to the relevant stakeholders. The stakeholders in our simple example are the software engineer and the test engineer. In the real world, end users or customers are often the ones to report defects. In this case the closed-loop change management process would need to ensure that the end user who submitted the defect is notified once the defect is resolved and, ideally, that the status information is communicated at appropriate points in the process.

However, just because defect-tracking systems can enable a closed-loop change management process doesn't mean they do so in the most productive way possible. It is true that they provide a single coordination point for all stakeholders, who in our example are the software engineer and the test engineer. But is coordination at the point of the tracking system itself the most productive method?

In actuality, productivity remains highest when stakeholders are working within their preferred environments, be they development tools or test management systems. Therefore, a better way to coordinate – and to enable the highest productivity – is to enable all stakeholders to access the defect-tracking system from within their preferred environments, while still ensuring that the change data is managed and protected with a single underlying system. Persona-driven solutions are one way to achieve this.

Persona-Driven Solutions

The basic concept of persona-driven solutions is to provide the end user with an interface that is tailored to meet the needs of the user and his or her role in the process.

The defect resolution process example has two primary personas involved: the software engineer, and the test engineer. Each persona has responsibility for different activities within the defect resolution process. In addition, each persona likely uses a different software application as their primary tool. A software engineer typically works within an integrated development environment (IDE), such as IBM WebSphere Studio Application

Developer. A test engineer often works within a test management tool, or may work primarily within the defect-tracking system. For our purposes, we'll assume that the test engineer is using a test management tool.

A simple solution is to provide different customized views within the defect-tracking system. A software engineer would be presented with information that is most relevant to the task of fixing the defect, and the test engineer would have his or her own personalized view. But this solution still requires that each persona switch between applications in order to complete the process. Why not provide the necessary functionality within the environment that is native to each persona?

This is exactly the goal of a persona-driven interface for the defect resolution process. For the software engineer, this solution provides an interface from within the IDE that presents the assigned defect as a new task within the IDE, lets the software engineer view the details of the defect, and then provides access to update the defect with notes, or with resolution information once the defect is fixed.

The same solution applies to the test engineer, but from within the test management tool. From within the test management tool, a persona-driven interface allows the submittal of a defect (including a system for automatically submitting defects if there is test automation), the ability to update a defect with additional notes and information, and assignment of the defect. Once the software engineer has fixed the defect, the process should support an automatic notification of the test engineer that a fix for the defect has occurred, and the validation process should begin. Once the defect is validated, the persona-driven interface will allow the test engineer to close the defect from within the test management tool. Let's look at the defect resolution workflow again, this time adding some detail (see Figure 2).

Providing a native interface from within the persona's primary application enhances coordination between the software engineer and the test engineer. The software engineer no longer needs to open the defect-tracking system to see what work they have, or receive yet another automatic e-mail notification that a new defect has been assigned. By allowing the software engineer to manage defect resolution activities within the IDE, the organization ensures that their process is being followed, and that defects are being resolved in a timely fashion. Similarly, the test engineer can submit defects directly from the test management tool, and be assured that the software engineer will be assigned the task. Once a fix is available, the test engineer will be automatically notified by an added validation task, seen from within the test management tool.

Often the process for resolving a defect isn't cut and dried, and you find an iterative process occurring. A software engineer may not be able to reproduce the problem and may request additional information from the test engineer. The test engineer may receive a fix that fails the validation test, sending the defect back to the software engineer. Imagine how increasingly inefficient this process becomes when each persona must constantly open or switch to a separate application – the defect management system – during these iterations. With persona-driven solutions these iterative steps are covered by properly – and automatically – communicating status to each party.

If the solution includes a defined workflow that provides more details of the "fix" activities performed by the software engineer, this solution would provide a status update as the different points in the process are achieved. For example, an organization may define a subprocess within the fix function: verify defect, identify module(s) to change, assess impact of change, apply coding change, functionally test fix, submit to QA for validation. The solution can be configured such that each of these states within the fix process is visible as a status within the defect-tracking system. In this way the test engineer could identify where the software engineer is in the process. In addition, an automatic escalation based on events or nonevents can be implemented – an important feature for organizations that have strict service-level agreements. The level of granularity in status communication is dependent upon the needs of the organization.

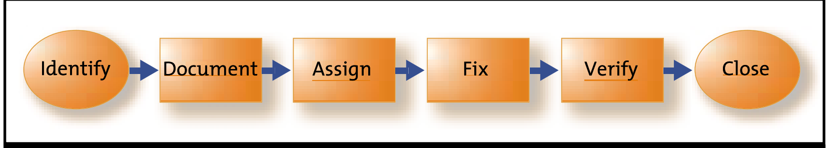


FIG. 1: DEFECT RESOLUTION PROCESS

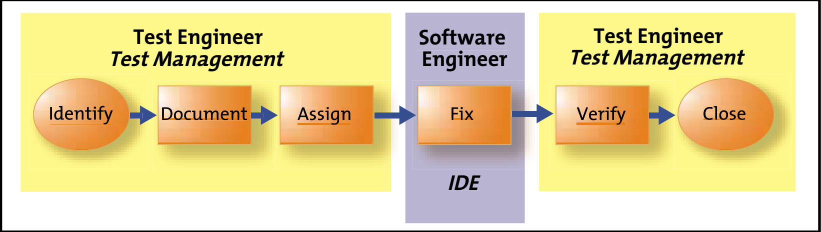


FIG. 2: PERSONA-DRIVEN DEFECT RESOLUTION PROCESS

Conclusion

All stakeholders benefit from a closed-loop change management system because it ensures that proper communication and coordination are automated, managed, and documented. Using a persona-driven solution ensures that closed-loop change management processes are also efficient and highly productive, leaving more time for everyone to focus on delivering a higher-quality software application with less time and risk.

Automated Error Prevention

Preventing errors in business systems

– BY ADAM KOLAWA –

ABOUT THE AUTHOR
Adam Kolawa is CEO, chairman, and a cofounder of Parasoft, a company that creates value-added products that significantly improve the software development process. Adam's years of experience with various software development processes has resulted in his unique insight into the high-tech industry and his uncanny ability to successfully identify technology trends. Adam is a well-known writer and speaker on industry issues and in 2001 was awarded the Los Angeles Ernst & Young Entrepreneur of the Year Award in the software category.

E-MAIL
ak@parasoft.com

Software errors cause not only system functionality problems, but also delayed releases, budget overruns, decreased development team productivity, and blemished corporate reputations. Errors are especially serious when they occur in code built upon WebSphere, which is typically at the core of critical business systems. If errors occur so close to the core of a business system, they will impact virtually every aspect of the system.

Automated Error Prevention (AEP) is a well-defined and highly accepted methodology for preventing errors. AEP helps you prevent errors in an automated fashion. This article describes how to apply AEP to WebSphere development.

Understanding AEP

AEP is a logical and practical application of W. Edwards Deming's principles of Total Quality Management, a process that revolutionized the manufacturing industry in the 1930s and that has been used to successfully improve quality and productivity in a number of industries ever since. Deming found that the key to improving quality and reducing inefficiency is to stop chasing after defects one by one, and instead prevent defects by improving the production line process that permits defects to be introduced. The following process improvement approach lies at the heart of AEP:

1. Detect an error.
2. Isolate the cause of the error.
3. Locate the point in the process that created the error.
4. Implement practices to prevent the error from recurring.
5. Monitor for improvements.

The greatest obstacle to implementing this process in the software industry is that many of the required tasks can be difficult, time-consuming, and expensive. However, it is now possible to automate most of the required tasks, thanks to recent developments in open source software tools, commercial software tools, and books/articles that explain how to build your own automation scripts or tools.

So how does AEP fit into the software life cycle? If you look at the software life cycle illustrated in Figure 1, you will see that the life cycle circulates through a number of stages. At each stage, a variety of different error prevention practices work together to stop errors. These practices include coding standards enforcement, unit testing, integration testing, load testing, connectivity verification, and monitoring. When one of these practices exposes an error, that error is analyzed, then the process that allowed the error is modified so that it prevents the same type of error from recurring.

This creates a feedback loop in which each error found serves as a catalyst for process improvement and error prevention. For example, assume that load testing reveals a performance problem caused by a critical piece of Java code that uses String instead of StringBuffer for nonconstant strings. Using AEP, you would not only correct this specific piece of code, but also implement and automatically enforce a new coding standard that forbids use of that inefficient coding construct. In this way, the AEP feedback loop helps prevent an entire class of performance problems.

Applying AEP During Business System Development

As Figure 1 shows, there are a variety of AEP practices that prevent and expose errors, which fuels the AEP feedback loop. This article focuses on the practices most critical to WebSphere development. For more information on the other AEP practices listed in Figure 1, visit www.parasoft.com/jsp/aep/aep.jsp.

Following WebSphere Coding Standards

You can prevent many errors by following recommended WebSphere coding standards as you write code for new functionality, customization, and integration. These standards help you avoid writing code that is technically valid and correct, but likely to cause functionality, perfor-

mance, or scalability problems on WebSphere Application Server. The following WebSphere coding standards are among those recommended in the IBM white paper "WebSphere Application Server Development Best Practices for Performance and Scalability."

- Release HttpSessions when finished.
- Minimize synchronization in servlets.
- Do not use SingleThreadModel.
- Reuse datasources for JDBC connections.
- Release JDBC resources when done.
- Minimize use of System.out.println.
- Avoid String concatenation "+=".
- Reuse EJB homes.
- Do not use Beans.instantiate() to create new bean instances.

The complete list of standards, as well as the justification for each standard, is available at www-3.ibm. com/ software/webservers/appserv/ws best practices.pdf.

Documenting Interfaces with Design by Contract

Another AEP practice, Design by Contract (DbC), prevents integration problems. Integration is an especially error-prone phase for business system projects. These projects typically involve a moderate amount of time developing code around the middleware – the heart of the system – then a great deal of time trying to ensure that the glue code, existing or purchased components, and other hardware and software interact successfully. Even if each part is solid in and of itself, problems in the interfaces between the components can cause system functionality problems – especially if the problem occurs in a critical or frequently used interface.

Design-by-contract involves formally documenting each unit's interface requirements and assumptions. When each unit's interfaces are clearly documented in a contract, developers are less likely to introduce errors by writing code that incorrectly interfaces with other units. In addition, if the code is compiled with a DbC-enabled compiler such as iContract or Parasoft's Jcontract, you receive automatic and immediate notification if interface problems arise during test executions. This allows you to perform an extensive amount of interface verification with a relatively minor investment of time and effort.

Figure 2 illustrates where contracts (represented by small squares) should be added to units (represented by circles). Ideally, contracts specify requirements that must be satisfied before the unit executes, after the unit executes, and (optionally) at various points during execution.

Building and Extending Test Cases

Another way to prevent errors in WebSphere is to create tests that expose errors at the unit, submodule, module, and system levels, and use those tests to build a comprehensive error prevention test suite. This test suite can automatically expose problems introduced by a system modification – even problems that are the result of strange, unexpected side effects.

One advantage of having such a test suite is that it allows you to expose each error as early as possible. Consequently, you can fix it when it is fastest, easiest, and cheapest to do so – as well as reduce the chance for it to cause additional bugs (from code interactions or code reuse).

An even greater benefit of this test suite is that it fuels the AEP feedback loop. Each time it exposes an error, it helps you identify an error-prone part of your development process. You can then prevent an entire class of errors by modifying the process so that it does not allow the same type of error to recur.

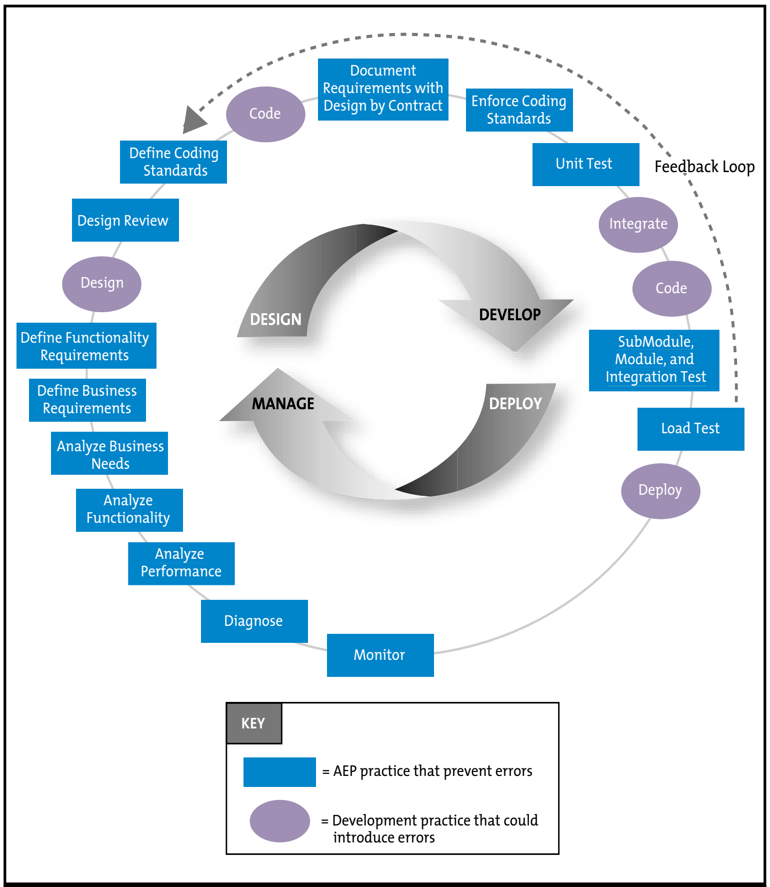

Document Requirements with Design by Contract
Enforce Coding Standards
Code
Define Coding Standards
Unit Test
Feedback Loop
Design Review
Integrate
Design
Code
DESIGN
DEVELOP
Define Functionality Requirements
SubModule, Module, and Integration Test
Define Business Requirements
MANAGE
DEPLOY
Load Test
Analyze Business Needs
Deploy
Analyze Functionality
Analyze Performance
Diagnose
Monitor
KEY
= AEP practice that prevent errors
= Development practice that could introduce errors

FIG. 1: INTEGRATING ERROR PREVENTION INTO THE FULL SOFTWARE LIFE CYCLE

As Figure 3 shows, the error prevention test suite creation process can be a natural progression that begins with building unit tests, then extending those tests so that they verify the submodule, the module, and then the entire system. You start by creating test cases that verify a unit (for example, a Java class). Once the unit is verified, you have a set of test cases that essentially freezes the correct functionality of the unit; if a change introduces an error into the unit, the test cases will expose it. Next, you extend the unit test cases so that they verify the submodule, then the module, then the entire system. Because you are extending and reusing test cases, you create a test suite that will automatically alert you if a modification introduces a problem into any aspect of the system.

For a more detailed look at how to develop such a test suite, consider an example. Assume we are developing a business system that provides our partners with a way to view inventory and order supplies. The partners access the application through a browser, and then beans access the database to perform the requested operations and deliver the results to the partners' browsers.

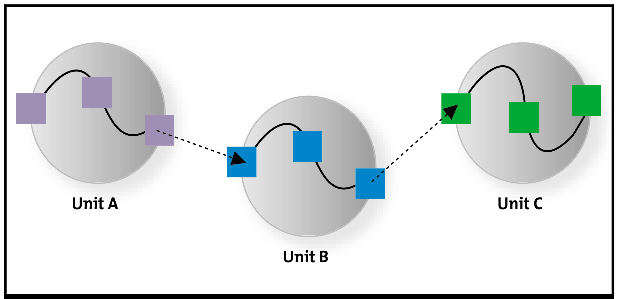


FIG. 2: ADDING CONTRACTS TO UNITS

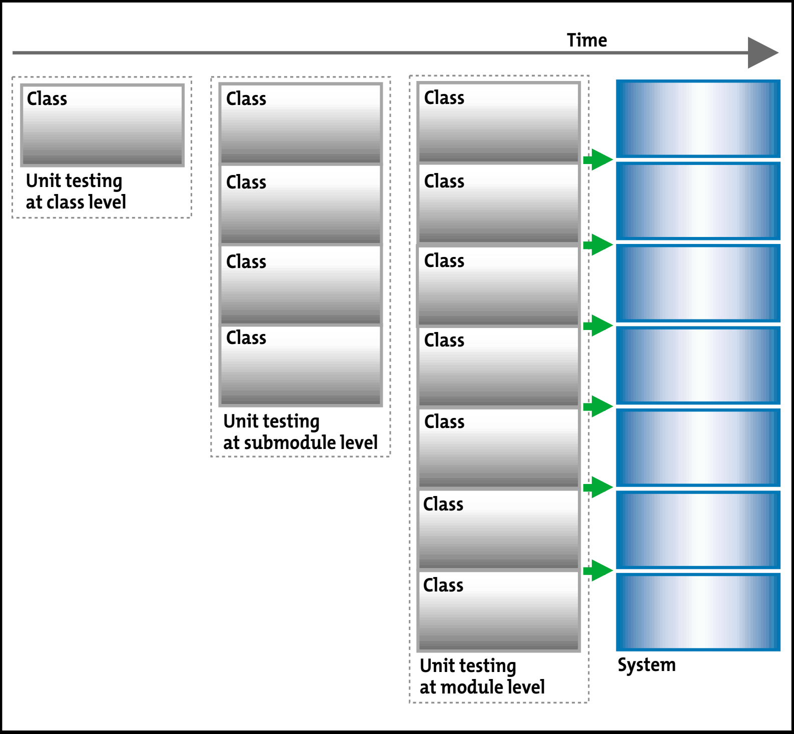

Time
Class
Unit testing at class level
Class
Class
Class
Class
Unit testing at submodule level
Class
Class
Class
Class
Class
Class
Class
Unit testing at module level
System

FIG. 3: THE NATURAL PROGRESSION FROM UNIT TESTS TO SYSTEM TESTS

The moment each bean is completed, we perform unit testing – including white-box and black-box testing – to verify that it is robust and correct. This involves creating a test harness and any necessary stubs for external resources, sending test inputs using a unit testing tool such as JUnit or Parasoft's Jtest, then verifying the outputs. This general unit testing process is illustrated in Figure 4. We perform this process for each bean we write. Because we have created test cases that freeze the correct behavior of each unit, we can instantly verify whether code modifications introduce unit-level problems by running the applicable tests. Thus, by collecting all of these test cases and running them on a regular basis (e.g., nightly), we establish the foundation of an error prevention test suite. In addition, the verified input/output relationships serve as known checkpoints that we can use to verify the success of our submodule, module, and system tests.

If any errors are exposed at the unit level, we not only correct the errors, but also use AEP to prevent classes of similar errors from recurring. For instance, if we found that an error occurred because we used assignment in an if statement condition when we should have used equality [i.e., we wrote if (a=b) when we should have written if (a==b)], we could create and enforce a coding standard that forbids the use of assignment in if statement conditions. We could then check this standard automatically as we write code to ensure that code containing this problem is never added to the source code repository and integrated into the system. This is just one of many possible ways that the AEP feedback loop can be used to prevent errors through process improvement.

We start extending the unit test cases when we have created enough beans to have a submodule. A submodule is a collection of units that are related, but not yet complete enough to be executed outside of a testing framework. As shown in Figure 5, the submodule in this example includes several related beans that are running on WebSphere, but are not yet accessible from a Web server. They will eventually connect to a database, but the database is not yet available.

To test this submodule, we combine and extend the available unit test cases. For instance, assume we have a set of unit test cases for Bean A, Bean B, and Bean C – three beans that work together. Each test case has a test input and an expected output. To build a submodule test case from available unit test cases, we use a tool such as JUnit to invoke Bean A with a unit test input, verify the output at Bean A's checkpoint, send that output to Bean B, verify the output at Bean B's checkpoint, send that output to Bean C, then verify the output at Bean C's checkpoint. Each resulting submodule test case is a combination of multiple unit test cases we previously created for each bean. We already know the correct reference values for each checkpoint as a result of our unit testing, so verifying the output at each step is simply a matter of checking whether the values achieved during the test execution match the expected values. In other words, we create new test cases by reusing and extending the checkpoints established during unit testing.

QUEST SOFTWARE

HTTP://JAVA.QUEST.COM/QCJ/WDJ

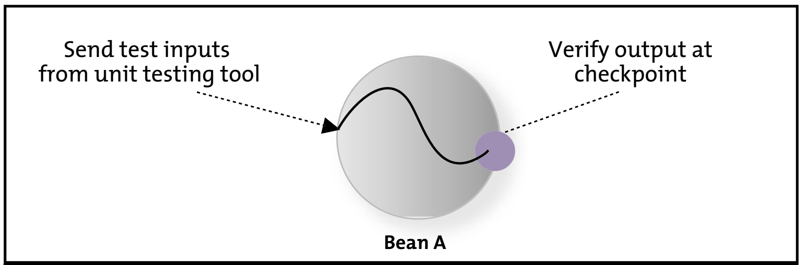

Send test inputs from unit testing tool
Verify output at checkpoint
Bean A

FIG. 4: TESTING A UNIT

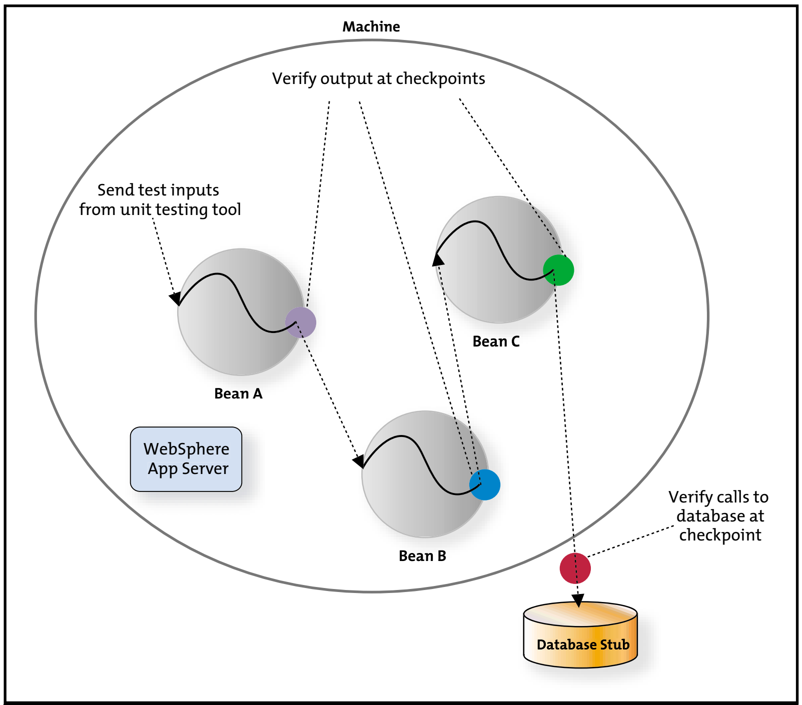

Machine
Verify output at checkpoints
Send test inputs from unit testing tool
Bean A
Bean B
Bean C
WebSphere App Server
Verify calls to database at checkpoint
Database Stub

FIG. 5: EXTENDING UNIT TESTS INTO SUBMODULE TESTS

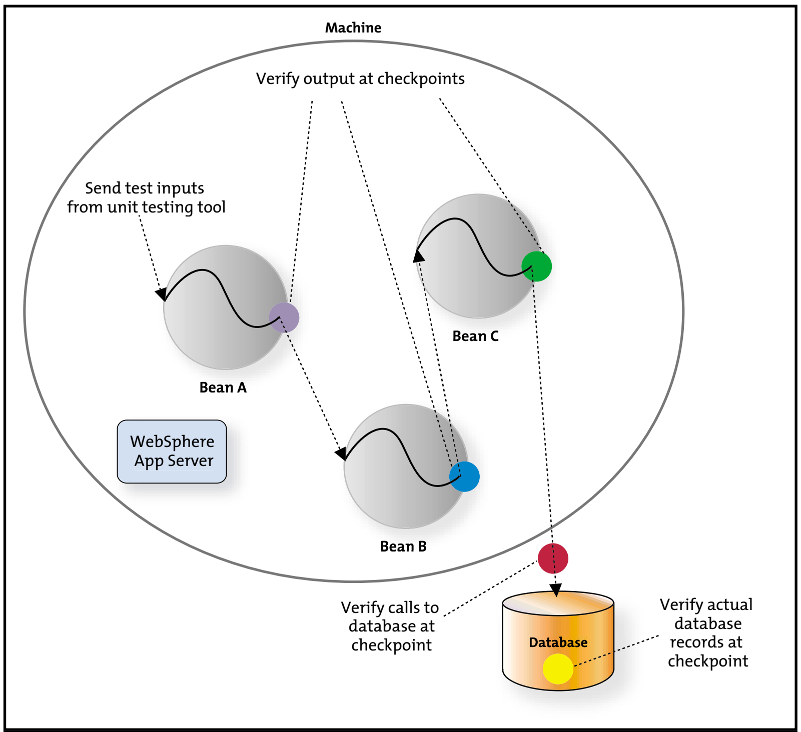

Machine
Verify output at checkpoints
Send test inputs from unit testing tool
Bean A
Bean B
Bean C
WebSphere App Server
Verify calls to database at checkpoint
Database
Verify actual database records at checkpoint

FIG. 6: EXTENDING SUBMODULE TESTS INTO MODULE TESTS

In addition, because the submodule is designed to send or retrieve data from the database, we want to verify whether that functionality operates correctly. The database is not yet available, but we can start verifying this functionality by configuring each submodule test case to write database calls to a file that serves as a database stub. This creates a new checkpoint that exposes incorrect database calls.

If submodule testing exposes an error, we correct the code, as well as apply AEP to determine which process allowed the error, then implement a new quality control measure that prevents the same type of error from recurring. For instance, assume that we find problems in the interface between units. If we did not already use design-by-contract to document and verify interfaces, we could improve our process by incorporating design-by-contract and then using automated tools to verify whether or not we were adding sufficient contracts to each unit. If we were already using design-by-contract, we could implement an additional process improvement measure, such as requiring additional verification checkpoints that compare actual data values to expected ones.

Once the database is connected to the submodule, we have a complete module and can start extending the submodule tests into module tests (see Figure 6). At this point, we have tests that span multiple units, and checkpoints at each unit as well as at the stubbed connection to the database. To extend these tests, we reuse the submodule tests that feed inputs through the units and check the values at the established checkpoints, and then we verify whether each of those tests performs the expected database operation. The database operation can be verified using SQL, a database verification tool such as DbUnit or Parasoft's DataRecon, or by checking the values collected by a JDBC sniffing tool.

If a module test exposes an error, we correct the code and implement AEP to prevent classes of similar errors from recurring. For instance, assume our tests reveal that application data is not being stored properly in the database. In this case, we would implement AEP by creating a range of test cases that verify whether code is correctly adding records to the database. We would also modify our testing process to require that each database-related test case explicitly verify whether the correct data is being stored in the database.

Once we can execute the system as a user would – in this case through a browser that interacts with our modules through a Web server – we can start extending our various module test cases into system test cases (see Figure 7). System-level tests should initiate and monitor simulated user transactions. These tests watch how a transaction travels through a system's interfaces, and then verify the result of those interactions. If the transaction functions correctly, all checkpoints will contain the expected values. Because each system-level transaction is a chain of complex interactions, I recommend that you focus on verifying one transaction at a time. A reasonable goal is to verify a represen-

DIRIG SOFTWARE

WWW.EBIZPERFORM.COM

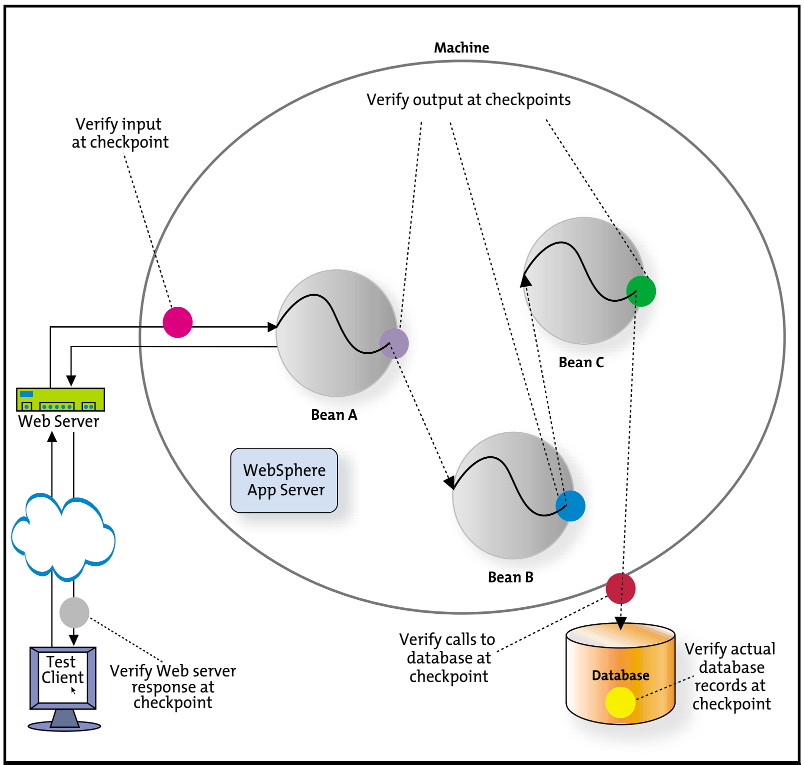

Machine
Verify output at checkpoints
Verify input at checkpoint
Bean A
Bean B
Bean C
Web Server
WebSphere App Server
Test Client
Verify Web server response at checkpoint
Verify calls to database at checkpoint
Database
Verify actual database records at checkpoint

FIG. 7: EXTENDING MODULE TESTS INTO SYSTEM TESTS

tative sample of the most critical and popular user transactions, one at a time. At the least, your test suite should exercise every system entry point that the user interface provides and exercise a representative sample of the possible input types for each entry point.

To create system-level test cases that check the established checkpoints as well as the interface with the user, we extend the beginning and end of our module tests. First, we configure a test client to invoke the module with the previously tested inputs. The test client can be a Web testing tool or an actual user exercising the application from a browser. Because the test is using the previously tested inputs, we can verify data values throughout the system by checking whether the actual value at each checkpoint corresponds to the expected value.

In addition to verifying the existing checkpoints, we create additional checkpoints – one that verifies whether the module was invoked appropriately, and one that verifies the response returned to the client. For this application, verifying the response involves checking the page sent to the client after each test action. Verification might include content verification, code validation, accessibility verification, and so on. Tools such as Watchfire's Bobby, the W3C validators, and Parasoft's WebKing can automate this verification process.

By extending the unit test cases to span the entire system, we establish a test suite that remembers the results of every checkpoint we ever verified, and instantly notifies us when a system modification introduces a problem into any part of the system. Many methods of system testing focus solely on verifying whether a user action produced the expected result, but this is not always an accurate assessment of whether the desired operation executed correctly. Consider a case in which the user action is supposed to record user information into the database, then send the user a confirmation message. If our system tests check only whether the user action produced the expected confirmation page, we could miss critical problems such as an interface problem that prevents the data from being recorded in the database.

The test suite we created, with its extensive system of checkpoints, instead targets that problem as soon as it occurs. The problem can then be fixed immediately, while the modification is still fresh in the developer's mind and before the error has a chance to spawn additional errors as a result of code interactions and reuse. More important, the process that allowed the error could be immediately repaired to prevent the introduction of similar database-application interface problems throughout this project and future projects.

The same general strategy can be applied to create system-wide test suites for business systems with different architectures. For example, if your module interacts with a legacy system or other resource outside of the submodule, you would verify that resource as we verified the database in the example. If your system is deployed as a Web service, you would exercise it using a test client, verify internal checkpoints throughout the system, then verify the HTTP traffic response to check the result. The verification logistics vary based on system architecture, but the effect is the same: a comprehensive test suite that automatically exposes errors as soon as possible.

Conclusion

With a comprehensive test suite that immediately and automatically exposes errors, you can apply AEP to prevent errors in two ways:

- You can identify and repair specific coding errors before they spawn additional errors as a result of code reuse and code interactions.
- You can identify and improve error-prone processes to prevent the introduction of additional errors.

Parasoft Automated Error Prevention Solutions

Parasoft Automated Error Prevention solutions provide a proven, cost-effective way to reduce the risk of business system failures by identifying and preventing errors. Solutions are available for Web applications, Web services, Java, C/C++, .NET, and databases. Each complete turnkey solution seamlessly incorporates error prevention into the full business system life cycle, and combines technologies and training to:

- Prevent development, deployment, and operational errors by automating practices such as coding standards enforcement, unit testing, functional testing, integration testing, load testing, security verification, and monitoring.
- Leverage work between development stages and development team members to reduce the overall amount of time and effort spent in production.
- Provide project management with the information required for managing the production, deployment, and maintenance of business systems.

VERSATA

WWW.VERSATA.COM/BUSINESSLOGICDESIGNER

Part 2: WSIF development

Web Services Invocation Framework

BY BORIS LUBLINSKY

The Web Services Invocation Framework (WSIF) is an architecture and programming model that – unlike today's most popular Web services APIs, JAX-RPC and JAXM – supports RPC and messaging invocation of Web services in a single programming model.

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. 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.

E-MAIL
boris.lublinsky@CNA.com

In Part 1 of this series I introduced WSIF and described its architecture and programming model. In this article I will discuss more advanced topics of WSIF programming, such as usage of different providers, JNDI bindings, asynchronous service invocation, and messaging.

Using the EJB Provider

After creating the basic Web service implementation, I attempted to use the same WSIF client code (see Listing 2) for EJB transport. Note: All of the code for this article can be downloaded from www.sys-con.com/websphere/sourcec.cfm. The only required change for switching from SOAP/HTTP communications to a straight EJB invocation was a change in the binding and service definition portions of the WSDL file (see Listing 1) as shown in Listing 3. This file uses WSDL extensions defined by WSIF. Extensions in the binding portion of the WSDL file are as follows:

- ***ejb:binding type:*** Denotes that the EJB provider should be used for service invocation.
- ***format:typeMapping:*** Defines mapping between SOAP definition (XML serialization types) and Java types. This clause is used for defining Java types/classes corresponding to both basic and complex data types.

Extensions in the service portion of the WSDL file include the ejb:address clause, which is used for specifying EJB location. This clause contains four major elements:

- ***className:*** Used for specifying fully the qualified name of the home interface of the EJB
- ***jndiName:*** Used for specifying the JNDI name of the EJB's home interface
- ***initialContextFactory:*** Used for specifying the fully qualified name of the JNDI initial context factory as required by initial context creation
- ***jndiProviderURL:*** Used for specifying the JNDI provider URL as required by initial context creation

After implementing changes to the WSDL as defined in Listing 3, the WSIF client code (see Listing 2) is run, producing the same result as execution of the simple WSIF client in Part 1:

```
  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
```

Using the Native JMS Provider

WSIF supports two types of JMS providers – SOAP over JMS using the Axis SOAP provider, and the native JMS provider. I evaluated the JMS provider first. Similar to the EJB provider, this file uses WSDL extensions defined in WSIF, as shown in Listing 4. The binding portion of the WSDL file uses the following extensions:

- ***jms:binding type:*** Defines that the native JMS provider should be used for service invocation. This binding type has two parameters.
  - Transport type: Used for service invocation. In this case transport type has to be http://schemas.xmlsoap.org/soap/jms.
  - Message type: Defines the JMS message type used for message invocation. The only supported message type is TextMessage, which corresponds to the JMS text message.
- ***format:typeMapping clause:*** Defines the serialization style used by the provider. The only supported

mapping type for this provider is encoding="XML".

Extensions in the service portion of the WSDL file include the jms:address clause, which specifies JMS addressing. This clause contains five required elements:

- ***jndiDestinationName:*** Used for specifying the JNDI name of the JMS destination object.
- ***destinationStyle:*** Used for specifying type of JMS destination object. Allowable values are “queue” and “topic”.
- ***jndiConnectionFactoryName:*** Used for specifying the JNDI name of the JMS connection factory object.
- ***initialContextFactory:*** Used for specifying the fully qualified name of the JNDI initial context factory as required by initial context creation.
- ***jndiProviderURL:*** Used for specifying the JNDI provider URL as required by initial context creation.

In addition to these, a number of additional JMS property elements can be specified. The one that I have used is JMSReplyTo property, which defines the JNDI name of the reply destination object and can be specified either in the WSDL file or programmatically through the message context, as follows:

```
  WSIFMessage context =
  operation.getContext();

context.setObjectPart(
WSIFConstants.CONTEXT_JMS_PR
EFIX +


"JMSReplyTo","jms\\mOutput")
;

operation.setContext(
context );
```

If this property is not set, and the JMS provider is used for request/reply interaction with the service, the provider will create a temporary queue for reply messages.

After implementing changes to the WSDL, as defined in Listing 4, the WSIF client code in Listing 2 was run, producing the following request message:

```
<?xml version="1.0"
  encod ing="UTF-8"?>
<addEntryRequest>
 <address>streetNum=1
streetName=The Waterfront
city=Some City state=NY
zip=47907

phoneNumber=areaCode=765
exchange=494
number=4900</address>
 <name>Jack</name>
</addEntryRequest>
```

The reason the message looks so strange is that for XML formatting to work correctly, special formatters have to be implemented for every Java class used in the service parameters. If the WSIF runtime cannot find these formatters, the content of each message part is stringified. Unless WSAD v5 Integration Edition is used, these files have to be written by hand. I’ve used WSAD v5 to generate two formatter files (see Listings 5 and 6), used for XML serialization of the Java classes Address and Phone.

The two formatters are loaded dynamically by the WSIF environment based on their fully qualified name. WSIF implementation assumes the following:

- The package name for the formatter class is the package name for the parameter class appended by the .jms.xml extension.
- The name of the file is the parameter class name appended by FormatHandler. After these classes were added, the WSIF client code was run, producing the XML request message shown in Listing 7.

Implementing Service for the Native JMS Provider

With the native XML encoding, WSIF allows for a very straightforward implementation of the service provider. A very simple provider implementation is presented in Listing 8. Most of the code implements a standard JMS listener. The WSIF-specific code starts in the onMessage method. The native JMS provider sets three JMS properties on the outgoing JMS message:

- ***WSDLOperation:*** Operation name, corresponding to the WSDL operation name
- ***WSDLInput:*** Name of the input message as defined in WSDL
- ***WSDLOutput:*** Name of the output message as defined in WSDL

In addition, if the service invocation is request/reply, the JMSReplyTo property is set by the provider with the destination object, to which the reply needs to be sent.

The onMessage implementation receives the incoming message and retrieves the parameters described earlier. Based on the value of the WSDLOperation property, the appropriate method is invoked. This method will process the request and produce a reply message, which is then sent back to the destination object. Before the message is sent back, the correlation ID of the outgoing method is set to the value of the ID of the initial message. This is required, because the service consumer is listening on the queue using a correlation ID–based selector.

The implementation of the method is very straightforward because WSIF implementation classes can be used for both processing incoming messages and building outgoing messages. In order to do this, the class constructor has to create the service and appropriate port exactly the way the client does. Based on the port, the service method gets a formatter, which allows unmarshaling of the incoming request and its conversion into a WSIF message. When this is done, Java objects can be extracted from the WSIF message the same way as with the client. After operations on data are executed, WSIFOperation can be instantiated, which allows the creation of an outgoing message that can be marshaled by the formatter class.

After implementing a service (see Listing 8), the WSIF client code was run again, producing the result shown earlier.

Using an Axis JMS Provider

As an alternative to the native JMS provider, an Axis JMS provider can be used. The advantage of this provider is that it sends true SOAP messages over JMS. In order to support the Axis JMS provider, binding and service definitions options for my original WSDL file have to be modified, as shown in Listing 9. This file uses WSDL extensions defined by WSIF. In the binding section I use soap:binding, the same as in the original file, but in addition, I specify http://schemas.xmlsoap.org/soap/jms as a transport.

Extensions in the service portion of the WSDL file include the jms:address clause, which is used to specify JMS addressing (including additional JMS properties), similar to the native JMS case. The only additional JMS property specified in this case is the SOAPAction property, which serves the same purpose as SOAPAction in the HTTP header. This parameter can be set in the JMS Axis implementation and used for any desired purpose. After implementing changes to the WSDL as defined in Listing 9, the WSIF client code was run, producing a true SOAP request message, as shown in Listing 10.

Implementing Service for the Axis JMS Provider

Creating a service implementation for the Axis JMS provider is more complex because WSIF does not provide the same level of support for the service implementation in this case that it does for the native JMS provider. For my implementation (see Listing 11) I used a direct invocation of the Axis engine from the JMS listener. In order to use this approach, a constructor of the JMS listener creates a configured instance of the Axis engine, which is responsible for routing of the request and invoking the service implementation (see Listing 12). Listing 13 shows the configuration file for AxisServer.

After implementing the service, the WSIF client code is run again, producing the same result as seen earlier.

Using JNDI to Store Service Information

As explained above, WSIF implementation relies on reading and processing WSDL files for building the runtime service model. In all of the examples discussed, I have used HTTP-based access to the WSDL documents. This approach forces client applications to hard-code exact locations (URLs) of the WSDL documents. This approach is not practical because in a real-world environment these files can move.

Alternative approaches for locating these files use either UDDI-based or JNDI-based registries. WSIF implementations provide built-in hooks for JNDI support. For JNDI-based access I used the JNDIHelper class, provided as part of WSIF distribution in the test/jndi directory. The code for binding WSDL to the JNDI and accessing service using JNDI is presented in Listings 14 and 15. One thing to keep in mind is that this code does not put WSDL into JNDI. The only thing that is stored in the JNDI is the WSDL location (URL), which means that the Web server that the JNDI information refers to has to be up and running for successful service invocation.

Setting Up Custom SOAP Headers Using WSIF

WSIF provides a very simple mechanism for implementing custom SOAP headers. I set custom headers using the Axis JMS provider; Listing 16 shows the code for populating of custom headers. WSIF allows populating both HTTP and SOAP headers through the operation context; SOAP headers for this operation have to be expressed as an ArrayList of XML elements. In my simple example, the headers were built by parsing the hard-coded sample XML snippet. In real-world applications these elements will be created dynamically using existing Java APIs for XML. After the code (see Listing 16) was added, the WSIF client was run again, producing a SOAP request message with custom headers, as shown in Listing 17.

Listing 18 presents a code snippet for extracting the SOAP headers from the reply. Again, the reply headers are extracted from the operation context. Because operation context is shared between request and response, SOAP headers for both will be in the context. The appropriate part name (org.apache.wsif.soap.ResponseHeaders) has to be specified for extracting the response SOAP headers.

Using WSIF for Sending XML Documents

WSIF provides a simple mechanism to take arbitrary XML documents as SOAP messages. I sent custom XML documents using the Axis JMS provider. This approach allows emulating basic behavior of the JAXM using WSIF APIs. In order to implement this scenario, I modified the original WSDL (see Listing 1) to:

- Eliminate all of the type definitions
- Modify message definitions to be single-part messages
- Change style to the document everywhere that RPC-style invocation was used
- Change encoding to literal
- Remove the encoding attribute from the SOAP body in the binding definitions

Listing 19 shows the resulting XML. In order to populate a message part with the XML document, a document tree element has to be set as a value of an appropriate body part (see Listing 20). Analogous to the custom headers scenario, discussed earlier, documents are built by parsing the hard-coded sample XML snippets. In real-world applications these documents will be created dynamically using existing Java APIs for XML. After the code (Listing 20) was added, the WSIF client was run, producing a SOAP request message as shown in Listing 21.

Implementing Message-Based Service

Service implementation in this case is very similar to the service implementation for the Axis JMS provider. In fact, I use the Axis engine for request routing in this case as well. Although the engine does not do much in this case, I decided to use it because of the request chains implementations provided.

Implementation of the JMS listener in this case (see Listing 21) is very similar to the Axis JMS provider implementation, with the difference that because document-style messaging does not contain any routing information I had to implement my own routing schema. I am using the SOAPAction parameter to define this information. Service implementation for this case is presented in Listing 22, and the Axis server configuration file is shown in Listing 23.

Asynchronous Service Invocation Using WSIF

It is often desirable to invoke a long-running service using the request/reply programming model. Many middleware products, including JMS, support an asynchronous request/reply programming model in which a requester sends a request, provides a callback object, and continues processing. When a reply comes back, a callback object is invoked to process the result of the service invocation.

In Web services programming practice, this model is often emulated through two one-way service invocations – one sending a request for service execution and another returning a reply. Although this approach works in principle, it puts an additional burden on the programmer to match replies to the original requests.

WSIF simplifies this programming model implementation by providing built-in support for asynchronous request/reply. In addition to the executeRequestResponseOperation method, which provides synchronous (blocking) operations invocation, the WSIFOperation class supports the executeRequestResponseAsync method, allowing for asynchronous (nonblocking) method invocation.

In order to implement asynchronous method invocations, two classes need to be implemented:

- ***Listener class:*** Allows listening for asynchronously delivered replies (see Listing 24)
- ***Callback class:*** Invoked when an asynchronous reply is delivered (see Listing 25)

With these two classes in place, built-in WSIF support is responsible for invoking the appropriate instance of the Callback class, making implementation of asynchronous request/reply very straightforward. Additional WSIF facilities allow for control of the asynchronous reply timeout.

Modification of the basic WSIF client, necessary for asynchronous request/reply implementation, is shown in Listing 26. After applying these changes and running the code, the same result, shown earlier, was produced.

Conclusion

The evaluation of WSIF presented in this article aims to better explain WSIF usage and to highlight some of its capabilities. WSIF is an extremely powerful set of Web services invocation APIs, supporting both RPC- and messaging-style invocation of Web services. A comparison of WSIF with other Web services invocation frameworks – JAX-RPC and JAXM – is presented in Table 1.

	JAXM	JAX-RPC	WSIF
Middleware model used	MOM	RPC	Hybrid
Service interaction models	Synchronous; asynchronous with the provider	One-way messaging and synchronous request/reply	One-way messaging, synchronous and asynchronous request/replies, notifications
Invocation models	Dynamic Invocation	Generated stub, dynamic proxies, dynamic interface invocation	Generated stub, dynamic interface invocation
Existing transport/ protocol support	SOAP/HTTP, SOAP/JMS, SOAP/SMTP	SOAP/HTTP	SOAP/HTTP, SOAP/JMS, JMS, SMTP, Java, EJB
Transport/protocol extensibility	Simple, through providers	Not defined	Simple, through providers
SOAP support	SOAP1.1, SOAP with attachments	SOAP1.1, SOAP with attachments	SOAP1.1
WSDL support	Non-supported	Supported only for one-way and synchronous request/ replies	Fully supported
Data typing	Not defined, any can be implemented through the message definition	Semantic messaging and strong typing	Semantic messaging and strong typing
Tooling options	None	Nice tools, but only for strongly typed synchronous request/replies or one-way communications.	WSDL2Java from Apache and WSAD 5 IE tooling from IBM, but only for strongly typed synchronous request/replies or one-way communications
Maturity/adoption level	Fairly mature, adoption is limited	Fairly mature, adoption is very high	Not very mature, adoption is limited

TABLE 1: COMPARISON OF WEB SERVICES INVOCATION FRAMEWORKS

Where Open Minds Meet

- **Learn** how companies have achieved higher profits and increased their productivity by utilizing Linux
- **Participate** in LinuxWorld's **world-class** education program and benefit from interactive training in the **all-new Hands-on Labs!**
- **Discover** the latest innovations and technologies from the hottest companies around
- **Hear** the latest developments and updates on the state of open source at our analyst roundtable discussion

CORNERSTONE SPONSOR

hp invent

PLATINUM SPONSORS

Computer Associates · DELL · IBM · intel · Novell · ORACLE · pogo [linux] Merging Business with Linux · redhat · Sun microsystems We make the net work. · VERITAS

GOLD SPONSOR

Hummingbird

SILVER SPONSOR

APPRO HPC Cluster Solutions

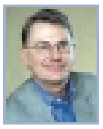
LINUXWORLD™ CONFERENCE & EXPO

Conference: August 4-7, 2003
Expo: August 5-7, 2003

The Moscone Center
San Francisco, CA

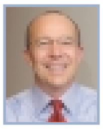
Join us this August and see why Linux is thriving! Government agencies and companies in the telecommunication, financial services, retail and manufacturing industries are turning to Linux to save money. Isn't it time you did? **LinuxWorld. Where Open Minds Meet.**

Attend Keynotes *and learn from inspiring visionaries who are devoted to Linux and open source.*



Tuesday, August 5th
10:30am-11:30am
Linux: The Next Step
Peter Blackmore
Executive Vice President
Enterprise Systems Group
Hewlett-Packard

Tuesday, August 5th
1:30pm-2:30pm
Jonathan Schwartz
Executive Vice President
Software Group
Sun Microsystems, Inc.



Tuesday, August 5th
4:30pm-5:30pm
Matthew Szulik
Chairman, Chief Executive Officer and President
Red Hat

Wednesday, August 6th
10:30am-11:30am
Linux and the Evolution of the Internet
Irving Wladawsky-Berger
General Manager
IBM Corporation

Wednesday, August 6th
3:30pm-4:30pm
Charles Rozwat
Executive Vice President
Server Technologies Division
Oracle Corporation

Special Presentation

Open to All Registered Attendees
Tuesday, August 5th
12:00pm-1:00pm

The Golden Penguin Bowl
Host: *Chris DiBona, Vice President – Marketing; Co-Founder, Damage Studios, Inc.*

Analyst Roundtable

Open to All Registered Attendees
Wednesday, August 6th
1:30pm-2:30pm

State of Open Source Roundtable
Moderator: *Larry Augustin, Partner, Azure Capital Partners*
Panelists: *Pierre Fricke, Executive Vice President of Web Application Infrastructure and Product Lifecycle Management (PLM) Infrastructure, D.H. Brown Associates, Inc.; Daniel Kusnetzky, Vice President, System Software Research, IDC; Ted Schadler, Principal Analyst in Software, Forrester; George Weiss, Vice President and Research Director, Gartner*

IDG WORLD EXPO

www.linuxworldexpo.com

Using a custom user registry in WebSphere Studio

Implementing J2EE Form-Based Authentication

BY GEETHA RAMASWAMY

The J2EE security model provides a built-in security concept based on roles provided by deployment descriptors. The deployment descriptors contain elements that allow us to map users or groups of users to specific roles (this information is used to authenticate the user) – and to specify which roles are allowed access to which resources. All users mapped to a particular role are allowed access to a certain resource. (This information is used to determine if the user is authorized to access the resource or not.)

ABOUT THE AUTHOR

Geetha Ramaswamy is a technical architect at SBC Communications Inc. She holds a master's degree in computer science and has five years of experience working with Java and Web technologies.

E-MAIL

g_ramaswamy@yahoo.com

For purposes of discussion, let us consider a simple J2EE application that has a few Web pages and one Enterprise JavaBean (EJB), which is a simple session bean. The J2EE containers that house these two different types of J2EE application components are:

1. ***Web container:*** Houses static HTML pages, servlets, JavaServer Pages, image files such as GIFs, etc.; its deployment descriptor is web.xml.
2. ***EJB container:*** Houses EJB components; its deployment descriptor is ejb-jar.xml.

Web Container Authentication Mechanisms

Since this J2EE application has a Web component, it makes sense to authenticate users from a Web page. The Web container provides the following authentication mechanisms that can be applied to this situation:

1. ***Basic:*** In this mechanism, when the user attempts to access a protected resource, the Web container checks whether the user has been authenticated. If not, the browser's built-in login screen pops up to prompt the user to enter the username and password for the Web container to perform authentication
2. ***Form-based:*** In this mechanism, when the user attempts to access a protected resource, the Web container checks whether the user has been authenticated. If not, an application-specific login screen is displayed that prompts the user to enter the username and password for the Web container to perform authentication.
3. ***HTTPS:*** This mechanism uses HTTP over Secure Sockets Layer (SSL).
4. ***Hybrid:*** In basic and form-based authentication, passwords are not protected adequately. A hybrid mechanism overcomes this by running basic and form-based mechanisms over SSL.

From the above list of available mechanisms we are interested solely in the form-based authentication mechanism, since this best fits our requirement for an application-specific login screen to perform authentication.

User Registry

Having decided on the authentication mechanism, we need to determine where the user information is stored. We could use databases, Lightweight Directory Access Protocol (LDAP), or even the operating system to store user information. In this case, we'll use a relational database to store the user information. Since there is no industry-wide standard on the exact database schema of a custom user registry, WebSphere seeks to circumvent this problem by providing an interface (com.ibm.websphere.security.CustomRegistry) that can be implemented.

This article seeks to provide a step-by-step guide to implementing form-based authentication for a simple J2EE application using a custom user registry in WebSphere Studio Application Developer v5.

Writing the Simple J2EE Application

PREREQUISITES

Start WebSphere Studio:

1. Go to the Windows Start menu.
2. Select Programs –> IBM WebSphere Studio –> Application Developer 5.0.

About the Application

We are about to create a simple J2EE application that manages the inventory for a retail store. Here we have two primary roles:

1. ***Manager:*** Denotes the role of the inventory manager. A user assigned to this role is allowed to order and view inventory.
2. ***Clerk:*** Denotes the role of inventory clerk. A user assigned to this role is allowed to view only inventory.

In order to illustrate these levels of security, we will implement the following:

1. For the Web component, we will create two files, order.html and view.html. In the Web deployment descriptor we will allow the "Manager" to access both files. However, we will allow the "Clerk" access to only view.html.
2. For the EJB component, we will create a session bean having two business methods, getOrder() and getView(). In the EJB deployment descriptor we will allow the "Manager" access to both of these business methods, but allow the "Clerk" access to only the getView() business method.

Custom User Registry Database

For implementing the custom user registry interface provided by WebSphere the database tables should follow a certain structure. The database scripts for Oracle are available in Listing 1. (All of the code listings for this article can be downloaded from www.sys-con.com/websphere/sourcec.cfm.) The CustomUserRegistry class described later in this article will access user information in these tables via JDBC to authenticate users.

Creating the Sample J2EE Application

Create a new J2EE 1.3 enterprise application project in the WebSphere Studio Workbench called FormAuthExample and click Finish (see Figure 1). Then perform the following steps:

1. Right-click on FormAuthExample-Web project.
2. Select Properties.
3. Click on Java Build Path.
4. From the Java Build Path option, click on the Projects tab. Check the box next to FormAuthExampleEJB project.
5. From the Java Build Path option, click on the Libraries tab. Click on the "Add External JARs" button.
6. Browse and select the following file : C:\Program Files\IBM\WebSphere Studio\wstools\ eclipse\plugins\ com.ibm.websphere.aes.v4_4.0.4\ lib\websphere.jar.
7. Click OK.

Configuring the Web Application

In the Web Perspective (FormAuth-ExampleWeb project), create the login.html page:

1. Right-click on /Web Content.
2. Select New–>HTML File.
3. Enter login.html in the Name field.
4. Click Finish.
5. Modify the login.html file with the code in Listing 2.

Next, create the index.html page:

1. Right-click on /Web Content.
2. Select New–>HTML File.
3. Enter index.html in the Name field.
4. Click Finish.
5. Modify the index.html file with the code in Listing 3.

Then create the error.jsp page:

1. Right-click on /Web Content.
2. Select New–>JSP File.
3. Enter error.jsp in the Name field.
4. Click Finish.
5. Modify the error.jsp file with the following code:

```
<html>
<head><title>Error
Page</title></head>
<body>
Error Page
</body>
</html>
```

Next, create the result.jsp page:

1. Right-click on /Web Content.
2. Select New–>JSP File.
3. Enter result.jsp in the Name field.
4. Click Finish.
5. Modify the BODY section of the result.jsp file with the following code:

```
<BODY>
<%
String result = (String)
session.getAttribute("result
");
if(result == null)
{
result = "Sorry, you do not
have the necessary privi-
leges to execute this busi-
ness method !";
}
%>
<%=result%>
</BODY>
```

Next, create a new folder in the Web Content directory named "secure":

1. Right-click on /Web Content.
2. Select New–>Folder.
3. Enter "secure" in the Folder Name field.
4. Click Finish.

Within the "secure" folder, create the order.html page:

1. Right-click on /Web Content/secure.
2. Select New–>HTML File.
3. Enter order.html in the Name field.
4. Click Finish.
5. Modify the BODY section of the order.html file with the following code:

```
<BODY>
<P>Order Inventory</P>
</BODY>
```

Within the "secure" folder, create the view.html page:

1. Right-click on /Web Content/ secure.
2. Select New–>HTML File.
3. Enter view.html in the Name field.
4. Click Finish.
5. Modify the BODY section of the view.html file with the following code:

```
<BODY>
<P>View Inventory</P>
</BODY>
```

Next, create a new package named "com.form.auth.example".

1. Right-click on /Java Source.
2. Select New–>Package.
3. Enter com.form.auth.example in the Name field.
4. Click Finish.

Within the com.form.auth.example package, create the FormAuthServlet.java class.

1. Right-click on /Java Source/ com.form.auth.example.
2. Select New–>Class.
3. Enter FormAuthServlet in the Name field.
4. Click Finish.
5. Modify FormAuthServlet.java with the code in Listing 4.

Within the same package, create the CustomUserRegistry.java class.

1. Right-click on /Java Source/com.form.auth.example.
2. Select New–>Class.
3. Enter CustomUserRegistry in the Name field.
4. Click Finish.
5. Modify CustomUserRegistry.java with the code shown in Listing 5.

Next, open the Web deployment descriptor, web.xml, located in /Web content/WEB-INF directory, and do the following:

1. Click on the Servlets tab. Add FormAuthServlet and its URL mapping (alias).
2. Click on the Security tab.
3. In the Security Roles section, add two roles, Manager and Clerk. *Note:* Role names are case sensitive. Also you must ensure that these role names are the same as those specified in the database table, GROUPS.
4. In the Security Constraints section (of the same tab), add the following security constraints:

```
/secure/order.html is acces-
  sible by the Manager role
  only
/secure/view.html is acces-
  sible by roles - Manager
  and Clerk
/FormAuthServlet is accessi-
  ble by roles - Manager and
  Clerk
```

5. Click on the Pages tab of the deployment descriptor. Select FORM as the authentication mechanism and enter FormAuthRealm as the Realm name, /login.html as the Login page, and /error.jsp as the Error page.

The source code for web.xml is shown in Listing 6.

Configuring the EJB Application

From the J2EE Perspective, create a new package, com.form.auth.example.ejb, in the ejbModule folder of the FormAuthExampleEJB project.

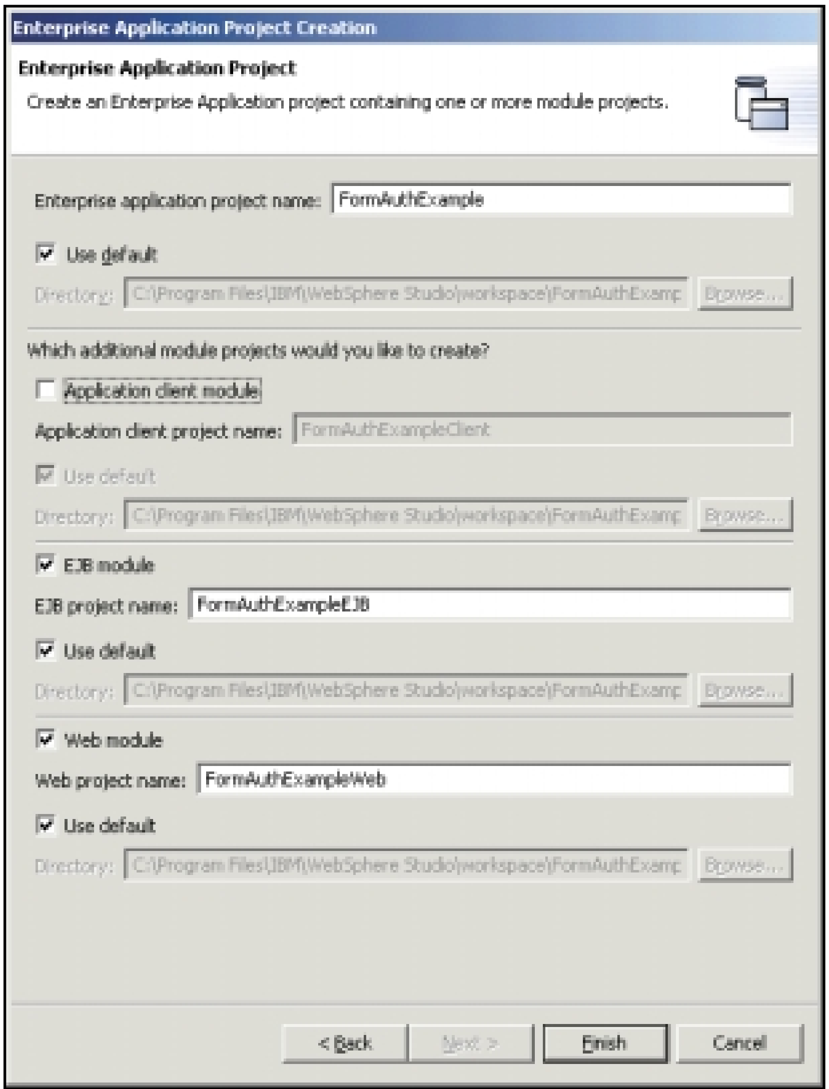
1. From the J2EE Navigator, right-click on /ejbModule/com.form.auth.example.ejb.
2. Select New–>Enterprise Bean. (EJB Type must be Session Bean.)
3. Enter FormAuthExampleEJB in the Bean Name field.
4. Click Finish.

Next, open FormAuthExampleEJBBean.java and add the following code:

```
public String getOrder() {
return "Order Inventory";
}

public String getView() {
return "View Inventory";
}
```

Promote these two business methods to the EJB's remote interface. After this, the FormAuthExampleEJB.java file should have the following signatures added to it:

```
public String getOrder()
throws
java.rmi.RemoteException;
public String getView()
throws
java.rmi.RemoteException;
```


Enterprise Application Project Creation
Enterprise Application Project
Create an Enterprise Application project containing one or more module projects.

Enterprise application project name: FormAuthExample

☑ Use default
Directory: C:\Program Files\IBM\WebSphere Studio\workspace\FormAuthExamp

Which additional module projects would you like to create?

☐ Application client module
Application client project name: FormAuthExampleClient
☑ Use default
Directory: C:\Program Files\IBM\WebSphere Studio\workspace\FormAuthExamp

☑ EJB module
EJB project name: FormAuthExampleEJB
☑ Use default
Directory: C:\Program Files\IBM\WebSphere Studio\workspace\FormAuthExamp

☑ Web module
Web project name: FormAuthExampleWeb
☑ Use default
Directory: C:\Program Files\IBM\WebSphere Studio\workspace\FormAuthExamp

FIG. 1: ENTERPRISE APPLICATION PROJECT CREATION WIZARD

Save and generate deployed code for the session EJB component.

Next, open the EJB deployment descriptor, ejb-jar.xml, which is located in the FormAuthExampleEJB/ejbModule/META-INF folder.

1. Click on the Assembly Descriptor tab.
2. Under Security Roles, add the two roles, Manager and Clerk.
3. Under Method Permissions, set the following permissions. The getOrder() method will be accessible only to the Manager role. The getView() method will be accessible to both Manager and Clerk.
4. Under Container Transactions, set the transaction type to Required for all the methods of the FormAuthExampleEJB bean.
5. Next, click on the References tab of the deployment descriptor.
6. Add a remote reference to FormAuthExampleEJB.
7. Add the security role references for Manager and Clerk. Save the EJB deployment descriptor and generate deployed code.

Next, open the EAR deployment descriptor or application.xml located in the FormAuthExample/META-INF folder.

1. Click on the Security tab.
2. Click on Gather, which "gets" all the roles. This will be evident because the two created roles will appear in the list box on this tab.
3. For each role, check the Users/Groups check box and enter the corresponding role's name as group name under the Groups list.

The source code for ejb-jar.xml is shown in Listing 7.

Configuring the WebSphere Studio Server Environment

1. Create a new server project named FormAuthExampleServer.
2. Within this project, create a new server and server configuration named FormAuthExampleServer for the test environment.
3. Add the FormAuthExample EAR to the FormAuthExampleServer configuration.
4. Double-click on the FormAuthExampleServer server configuration.
5. In the server configuration, click on the Configuration tab. Check the Enable administrative console check box. Click on the Paths tab and add the following paths under ws.ext.dirs:

```
C:\Program Files\IBM\
  WebSphere Studio\work-
  space\FormAuthExampleWeb
  \Web Content\WEB-INF\class-
  es
C:\Program Files\IBM\
  WebSphere Studio\work-
  space\FormAuthExampleEJB\
  ejbModule
```

6. Under Classpath, add:

```
C:\oracle\ora81\jdbc\lib\
classes12.zip
```

7. Click on the Environment tab. Under system properties, add the following:

```
driver = oracle.jdbc.pool.
  OracleConnectionPoolData-
  Source
url = jdbc:oracle:thin:
  @servername:1521:ORCL
  user = sample (Name of the
  schema)
password = ****** (DB
  password)
```

These system properties are used by the com.form.auth.example.CustomUserRegistry class

8. In the Security tab, check the Enforce Java 2 Security check box. *Note:* There are other settings in this tab that will need to be set after you complete the next section. Save the configuration.

Configuring the WebSphere Studio Admin Console

1. Start the FormAuthExampleServer server.
2. Once the server is started successfully, right-click on the server and select – Run administrative console. *Note:* Global Security must be disabled in order to run the administrative console. Therefore the Enable security check box in the Security tab of the server configuration must remain unchecked.
3. Type in any user ID and log in to the admin console. This user ID is used only for logging purposes.
4. Once logged in, using the left navigation tree of the admin console, click on Security–>User Registries–>Custom.
5. Specify the Server User ID as "geethar".
6. Specify the Server password as "pwd". (The user ID and password were inserted into the database tables as shown in Listing 1, which contains scripts to insert sample data after the tables are created.)
7. Specify the Custom registry class name as com.form.auth.example.CustomUserRegistry.
8. Click on the Custom Properties link on the same screen and enter the following values:

```
driver = oracle.jdbc.pool.
  OracleConnectionPoolData-
  Source
url = jdbc:oracle:thin:
  @servername:1521:ORCL
user = sample (Name of the
  schema)
password = ****** (DB
  password)
```

9. Save the configuration
10. Next, using the left navigation tree of the admin console, click on Security–>Global Security.
11. Check the Enable security check box.
12. Check the Enable Java 2 security check box.
13. Select Custom as the Active User Registry
14. Save the configuration.
15. Next, using the left navigation tree of the admin console, click on Security–>JAAS–>J2C Security.
16. Add a new entry whose:

```
Alias = ORCL
User ID = sample
Password = ****** (DB
Password)
```

17. Save the master configuration.

Ensure that the module dependencies are specified correctly in the Properties–>Project References for each project. For the FormAuthExampleWeb project, the projects FormAuthExample and FormAuthExampleEJB must be checked. For the FormAuthExample project, the projects FormAuthExampleWeb and FormAuthExampleEJB must be checked. For the FormAuthExampleEJB project, the project FormAuthExample must be checked.

Rebuild all three projects and publish to FormAuthExampleServer.

Restart the server. Test the application by running index.html on the server. When the user logs in as a Clerk, he or she can access only the View Inventory links, whereas when logging in as a Manager, the user can access all four links displayed on this page.

Conclusion

In this article, I have tried to demonstrate in a step-by-step manner how to implement form-based authentication using a custom user registry in WebSphere Studio Application Developer v5. This form of declarative security – being more flexible – is preferred over programming or encoding security directly into applications. However, in cases where a more fine-grained level of security is required, some programmatic security will need to be implemented in conjunction with declarative security to achieve the desired level of security.

Resources

- Heijmans, M. (2003). "An LTPA Custom User Registry." *WebSphere Developer's Journal*. SYS-CON Media. Vol. 2, issue 2.
- Subrahmanyam. (2000). "Java Servlets Advanced Features." *Java Developer's Journal*. Vol. 5, issue 2.
- Mahapatra, S. (2001). "J2EE Application Security Model." *Java Developer's Journal*. Vol. 6, issue 8.

LISTING 1

```
CREATE TABLE USERS
(
  ID      NUMBER(8) NOT NULL,
  USERNAME          VARCHAR2(20) NOT NULL,
  PASSWORD          VARCHAR2(20) NOT NULL,
  FIRSTNAME         VARCHAR2(20) NOT NULL,
  LASTNAME          VARCHAR2(20) NOT NULL,
  CONSTRAINT PK_USER PRIMARY KEY (ID )
);

CREATE TABLE GROUPS
(
  ID      NUMBER(8) NOT NULL,
  GROUPNAME         VARCHAR2(20) UNIQUE NOT NULL,
  CONSTRAINT PK_GROUP PRIMARY KEY (ID )
);

CREATE TABLE MEMBER
(
  USERID  NUMBER(8),
  GROUPID NUMBER(8),
  CONSTRAINT PK_MEMBER PRIMARY KEY (USERID, GROUPID )
);

ALTER TABLE MEMBER
 ADD CONSTRAINT MEMBER_USERS
 FOREIGN KEY (USERID)
 REFERENCES USERS (ID);


ALTER TABLE MEMBER
 ADD CONSTRAINT MEMBER_GROUPS
 FOREIGN KEY (GROUPID)
 REFERENCES GROUPS (ID);

 INSERT INTO USERS (ID, USERNAME, PASSWORD, FIRSTNAME,
 LASTNAME) VALUES (1, 'gbush','laura','George','Bush');

 INSERT INTO USERS (ID, USERNAME, PASSWORD, FIRSTNAME,
 LASTNAME) VALUES (2, 'bclinton','hillary','Bill','Clinton');

 INSERT INTO GROUPS(ID, GROUPNAME) VALUES (1, 'Manager');

 INSERT INTO GROUPS(ID, GROUPNAME) VALUES (2, 'Clerk');

 INSERT INTO MEMBER(USERID, GROUPID) VALUES(1,1);

 INSERT INTO MEMBER(USERID, GROUPID) VALUES(2,2);

 INSERT INTO USERS (ID, USERNAME, PASSWORD, FIRSTNAME,
 LASTNAME) VALUES (3, 'geethar','pwd','Geetha','Ramaswamy');

 INSERT INTO MEMBER(USERID, GROUPID) VALUES(3,1);
```

LISTING 2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Studio">
<TITLE>login.html</TITLE>
</HEAD>
<BODY>

<center>

<font face="arial" size=1 color="blue"> Please enter the
following information: </font>

<br>
<form method=POST action="j_security_check">

<font face="arial" size=1> Username <input type=text
name="j_username" size=20>
</font>
<br>
<font face="arial" size=1> Password <input type=password
name="j_password" size=20>
</font>
<br>
<input type=submit name=action value="Login">

</form>
<hr>
</center>
</BODY>
</HTML>
```

LISTING 3

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<META name="GENERATOR" content="IBM WebSphere Studio">
<TITLE>index.html</TITLE>
</HEAD>
<BODY>
<BODY>
<H2>Web Resources:</H2>
<A HREF="secure/view.html">View Inventory</A><BR/>
<A HREF="secure/order.html">Order Inventory</A><BR/>
<H2>EJB's:</H2>

<A HREF="FormAuthServlet?role=clerk">View Inventory</A><BR/>
<A HREF="FormAuthServlet?role=manager">Order
Inventory</A><BR/>
</BODY>
</HTML>
```

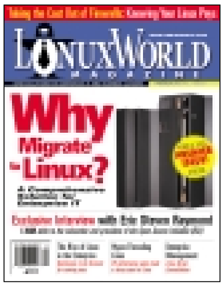
CLICK HERE TO DOWNLOAD THE CODE!
www.sys-con.com/websphere

A LIMITED TIME SAVINGS OFFER FROM SYS-CON Media

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS



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

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

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

☐ **Linux World Magazine**

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

☐ **Java Developer's Journal**

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

☐ **Web Services Journal**

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

☐ **.NET Developer's Journal**

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

☐ **XML-Journal**

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

☐ **WebLogic Developer's Journal**

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

☐ **ColdFusion Developer's Journal**

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

☐ **Wireless Business & Technology**

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

☐ **WebSphere Developer's Journal**

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

☐ **PowerBuilder Developer's Journal**

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

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

3-Pack

Pick any 3 of our magazines and save up to $275.00
Pay only $175 for a 1 year subscription plus a FREE CD

- 2 Year – $299.00
- Canada/Mexico – $245.00
- International – $315.00

6-Pack

Pick any 6 of our magazines and save up to $350.00
Pay only $395 for a 1 year subscription plus 2 FREE CDs

- 2 Year – $669.00
- Canada/Mexico – $555.00
- International – $710.00

9-Pack

Pick 9 of our magazines and save up to $400.00
Pay only $495 for a 1 year subscription plus 3 FREE CDs

- 2 Year – $839.00
- Canada/Mexico – $695.00
- International – $890.00

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

SYS-CON MEDIA

WSAD/ORACLE TUTORIAL

Six steps is all it takes

Running WSAD and WAS with Oracle

BY TROY HOLMES

ABOUT THE AUTHOR
Troy Holmes has been working in the IT industry for 14 years and is currently a J2EE architect using WebSphere 5.0. He has completed several large-scale J2EE applications using both BEA and WebSphere products. Troy is a certified Java programmer who has been working in the Java environment for five years. He also has more than five years of experience with Oracle and two years of experience with Informix.

E-MAIL
troy.holmes@prizum.com

During a recent EAI project, it came to my attention that it is difficult to find documentation on the topic of installing Oracle into WebSphere Studio Application Developer (WSAD) and WebSphere Application Server (WAS) 5.0. This article will attempt to expedite the search for this information by discussing both installation processes.

WSAD

STEP 1: INSTALL ORACLE

The first step is to install the Oracle drivers and the tnsnames.ora file into our system. To do this, install the Oracle client onto the machine that contains WSAD by following the instructions provided by Oracle. For simplicity's sake, this article assumes use of the defaults provided by the Oracle installation wizard.

STEP 2: SET UP THE DEFAULT USER PASSWORD

This is part of the new J2C security configuration and it is located on the Security tab. In this section, we will create a user ID and password and assign it to an alias ID. This alias is required in Step 5.

- Select the Security tab. It is located three tabs to the right of the Data Source tab.
- Next, add a JAAS Authentication Entry by selecting the Add button next to JAAS Authentication Entries.
- Fill in the following information and select the OK button:
  –Alias: OracleUser
  –User ID: The Oracle-defined user ID
  –Password: The password for the Oracle user
  –Description: Default Oracle user

STEP 3: CREATE A SERVER AND CONFIGURATION

WSAD uses an embedded server to test development code. To use this server we must first create a new server and configuration. Create a new server by selecting New –> Other –> Server/Configuration from the menu.

On the first screen enter Test Server as the server name and select Test Environment as the server type. Use the default port of 9080 and click on the Finish button. This will create a new server and configuration in the Server Configuration screen located in the lower panel of WSAD.

STEP 4: ADD THE JDBC DRIVER

This step will link the Oracle driver we installed in Step 1 to the newly created server/configuration.

- First, select the server and select the Data Source tab. This tab sets the configuration on the server.
- From the Data Source tab of the server configuration, select the Add button. This will display a list of databases.
- From this list, select the Oracle database; this will populate the provider types.
- From the provider types list, select JDBC:Thin:Driver.
- Select the Next button and assign the name OracleThinDriver to this driver.
- Finally, select the Finish button.

STEP 5: ADD THE DATA SOURCE

The data source is where we assign the JNDI name and alias users.

- First, highlight the JDBC provider that we created in Step 3 and select the Add button.
- A popup window will appear that requires the selection of a type of driver; here again, select JDBC:Thin: Driver.
- We have two options available on this screen, enabling us to select a version 5.0 data source or a version 4.0 data source. In this article, we are concerned only with new functionality and therefore will not be discussing the 4.0 setup. Select the 5.0 data source and then click the Next button.
- Leave the name and JNDI at their default values.
- Select the Alias list box and choose the alias created in Step 2, OracleUser. Add this alias to both types of authentication.
- Finally, select the Finish button.

STEP 6: EDIT THE RESOURCE

The only required field in the Resources screen is the URL. This field is used by the server to look up the tnsnames.ora file and find the port to the database. Select the URL and make the following modification.

The URL format is jdbc:oracle:thin: @xxx.xxx.xxx.xxx:1521:dbalias. This URL is broken into four sections delimited by colons. The URL is defined as follows:

- Provider type
- Oracle host IP address
- Oracle listener port
- Oracle database name

This URL needs to match the Oracle configuration for your installation. It is essential that these fields match the corresponding tnsnames.ora file. If any of the information is incorrect, you'll receive errors.

Figure 1 is an example of the completed data source.

Now that we've configured our WSAD environment, we can proceed to testing. This can be completed in six steps:

1. Create a new Web project.
2. Copy the servlet code from Listing 1 into your new Web project. Modify the table name and column name in the servlet code to match the table and columns in your database.
3. Modify the user ID and password to match your database.
4. Modify your web.xml to initialize the servlet on startup.
5. Publish the project to the server/ configuration we created earlier.
6. Start the server.

If successful, the server console will now display the output of our table.

WAS

The Admin Console is now run through the Deployment Manager. We will not go into detail on this process, as that is out of scope for this article. However, it is important to understand that all configuration management is handled via the Deployment Manager. If you would like more information on this topic, refer to the IBM Redbook (SG24-6195-00), titled "IBM WebSphere Version 5.0 System Management and Configuration."

STEP 1: INSTALL ORACLE

We must install the Oracle drivers on each application server. The Oracle driver is a server resource and the Deployment Manager does not manage it. To do this, install the Oracle client onto the machine that contains WAS by following the instructions provided by Oracle. As before, use the defaults provided by the Oracle installation wizard.

STEP 2: CREATE THE J2C DEFAULT USER ID AND PASSWORD

Now we need to set up our alias, which will be required in Step 5.

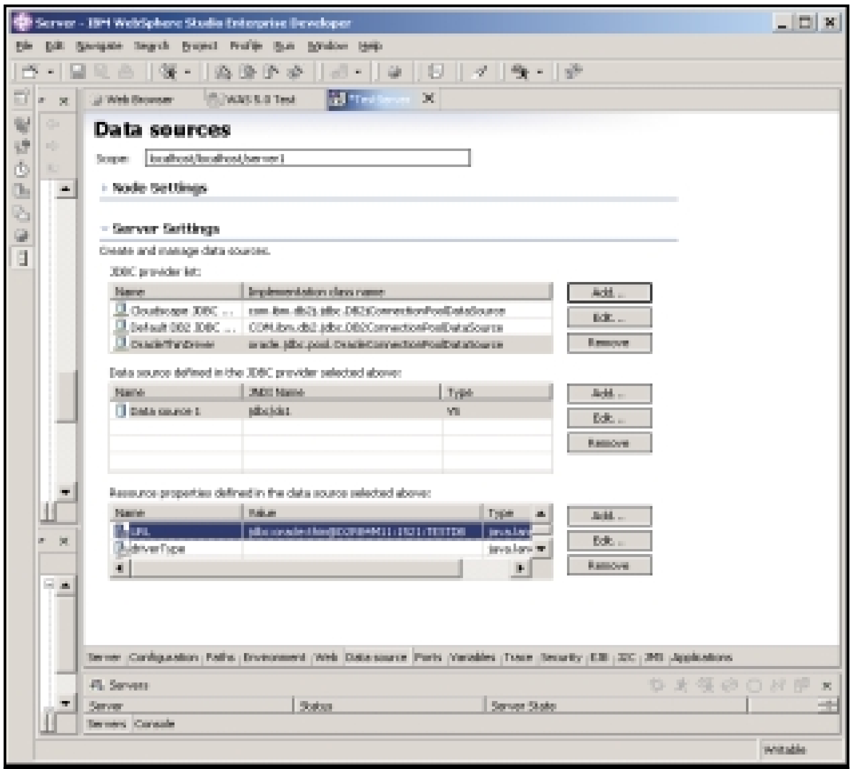
- Start the application server and deployment manager, and log in to the Admin Console.
- Select the Security link.
- Next, select the JAAS Configuration link. A list of menu options should be displayed.
- Select the J2C Authentication Data link, which will pop up a screen that displays the J2C authentication data entries.
- Select the New button and a window will display that enables the input of the alias information. Fill in the following information and select the OK button:
  -Alias: OracleUser
  -User ID: The user ID that you have set up in Oracle
  -Password: The password that you have set up in Oracle for the user ID above
  -Description: Default Oracle user

STEP 3: ADD THE JDBC DRIVER

- Expand the Resources tree and select the JDBC providers. This brings up the screen that requires the selection of a scope. You must select a scope and then the Apply button. If you do not select the Apply button, none of the modifications made in the next few steps will be propagated to the correct scope.
- In this exercise, we will propagate the changes at the cell level, which implies that the changes propagate to all nodes and servers within the given cluster. Select the Apply button and then the New button.
- A screen will appear that displays a list of JDBC providers. Select the Oracle JDBC Thin Driver and then the OK button.
- On the next screen, we will accept the defaults values; notice that the classpath has a path variable for Oracle. We will set up this variable in Step 5.
- Finally, select the OK button.

STEP 4: ADD THE DATA SOURCE

- Select the JDBC driver from the list of installed drivers.
- Select the Data Source link at the bottom of the page. Select the version 5.0 data source. This is marked as "Data Source" on the screen, while the version 4 data source is marked as "Data Source (version 4.0)."


Server - IBM WebSphere Studio Enterprise Developer
File Edit Navigate Search Project Profile Run Window Help
Web Browser
WAS 5.0 Test
Test Server
Servers Console

Data sources

Scope: localhost/localhost/server1

Node Settings

Server Settings

Create and manage data sources.

JDBC provider list:

Name	Implementation class name
Cloudscape JDBC ...	com.ibm.db2j.jdbc.DB2jConnectionPoolDataSource
Default DB2 JDBC ...	COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
OracleThinDriver	oracle.jdbc.pool.OracleConnectionPoolDataSource

Data source defined in the JDBC provider selected above:

Name	JNDI Name	Type
Data source 1	jdbc/ds1	V5

Resource properties defined in the data source selected above:

Name	Value	Type
URL	jdbc:oracle:thin:@[illegible]:1521:TESTDB	java.lan
driverType		java.lan

FIG. 1: COMPLETED DATA SOURCE

- Next, assign the JNDI name to the data source. Click the New button and a screen will appear that requires the input of the name and JNDI name. The application server does not put any default values into these fields. We will enter the same values that were the defaults in the WSAD example. For name enter data source 1, and for JNDI name enter jdbc/ds1.
- Select the list box on Container Managed and select the alias created in Step 2.
- Press the OK button

STEP 5: ADD THE RESOURCE

- Select the Data Source link. This will display the data source panel.
- Scroll to the bottom of the screen and select the Custom properties link.
- The only required field is the URL; therefore, select the URL from the Name column.
- The URL format is: jdbc:oracle: thin:@xxx.xxx.xxx.xxx:1521:dbalias. This URL should be defined in the same manner as in WSAD Step 6.
- Input the correct data into the URL value field and select the OK button.
- Select the Save option in the upper right-hand corner to save the configuration to the master configuration file. This will propagate the configuration to the nodes within the cluster

STEP 6: CREATE THE JDBC DRIVER ENVIRONMENT VARIABLE

The final step is to create an environment variable that enables WAS to access the JDBC driver.

- First, select the Environment link on the left toolbar.
- Next, select the Manage WebSphere Variables link. This will display a list of variables managed by WebSphere. We must remember to select the scope of the change as we did for setting up the data source.
- Select cell level and press the Apply button.
- Select the ORACLE_JDBC_DRIVER_PATH.
- Modify the value to correspond to the location of your Oracle driver. The default is "c:\oracle\ora81\jdbc\lib".
- Select the Save button in the upper right-hand corner.

Now that we have configured our WAS environment, we need to test our configuration. This can be completed in six easy steps:

1. Create a new Web project.
2. Copy the servlet code as before into your new Web project. Modify the table name and column name in the servlet code to match the table and columns in your database.
3. Modify your web.xml to initialize the servlet on startup.
4. Create a WAR file wrapping the servlet code for deployment.
5. Install the WAR file into WAS as a new enterprise application.
6. Start the application.

If we were successful, the log files of your server console should display our table information.

LISTING 1

```
// Servlet Code

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.naming.NamingException;
import javax.naming.InitialContext;
import javax.sql.*;
import java.sql.*;
/**
 * <strong>DBServlet</strong> retrieves the data from the
 * database used for testing connectivity
 *
 * @author Troy E. Holmes
 * @version $Revision:  1.4  $ $Date:   26 May 2003
 * 10:35:42  $
 */

public final class DBServlet extends HttpServlet {

  // ------------------------------------Instance Variables

  /**
   * The debugging detail level for this servlet.
   */

private int debug = 1;

// ---------------------------------------HttpServlet Methods

  /**
   * Gracefully shut down this servlet, releasing any
   * resources that were allocated at initialization.

*/
  public void destroy() {

         if (debug >= 1)
                 System.out.println("Finalizing servlet");
}

  /**
   * Initialize this servlet
   * @exception ServletException if we cannot configure
   * ourselves correctly
   */

  public void init() throws ServletException {

         // Process our servlet initialization parameters
         String value;

         value = getServletConfig().getInitParameter
         ("debug");

         try {

debug = Integer.parseInt(value);
         } catch (Throwable t) {
                 debug = 0;
         }

         // Load our data from persistent storage and log
         // to console
         try {

                 retrieve();

         } catch (Exception e) {
                 System.out.println("retreive exception
                 logged");

         }
  }

  // ---------------------------------------Public Methods

  /**
  * Return the debugging detail level for this servlet.
  */
```

-continued on page 43

SCO Versus IBM: The (Tangled) Story So Far...

WSDJ is not the appropriate forum for a discussion of the merits, or absence thereof, of one software company suing another at all, let alone for IP violations, which are notoriously difficult to prove or defend, or for damages running into literally billions and billions of dollars. Nevertheless, we will lay down the essence of what has happened at the time of this writing.

The easiest place to start is, unusually, in the middle. On June 16, 2003, the SCO Group went ahead with its threat to terminate IBM's right to use and distribute AIX software and filed for a permanent injunction requiring IBM to cease and desist all use and distribution of AIX and to destroy or return all copies of the Unix System V source code.

That sums up the entire case. The SCO Group, as the current owner of the license granted under the original 1985 Unix Software and Sublicensing Agreements between IBM and AT&T, had notified IBM on March 6 that it intended to terminate in 100 days if IBM did not correct certain actions that – according to the SCO – violate the agreement.

IBM refutes such claims categorically, and through Trink Guarino, director of IBM Media Relations, issued the following statement that same day:

Since filing a lawsuit against IBM, SCO has made public statements and accusations about IBM's Unix license and about Linux in an apparent attempt to create fear, uncertainty, and doubt among IBM's customers and the open source community.

IBM's Unix license is irrevocable, perpetual, and fully paid up. It cannot be terminated. This matter will eventually be resolved in the normal legal process.

IBM will continue to ship, support, and develop AIX, which represents years of IBM innovation, hundreds of millions of dollars of investment, and many patents. As always, IBM will stand behind our products and our customers.

Since then, more accusations have flown. *WSDJ* takes no position but must add that SCO VP Chris Sontag told Byte.com that, in his view, IBM's Linux efforts are equivalent to selling arms to terrorists. Why? Because Linux, he says, is used by terrorists. In other words, Sontag apparently expects the U.S. government to support his case against IBM and Linux as part of the war on terror.

Sontag has also accused Intel of violating U.S. export control laws banning weapons exports to North Korea by enabling Syria, Libya, and North Korea to build supercomputers with Linux – combined with inexpensive Intel hardware.

REPORTINGENGINES PRODUCT FULLY LEVERAGES JAVA PLATFORM

(Overland Park, KS) – Reporting Engines, a division of Actuate Corporation and a provider of embedded reporting solutions for J2EE Web and application servers, has announced the immediate availability of the Formula One product line. The Formula One e.Report Engine and e.Spreadsheet Engine offer developers the first full-featured, reporting toolset that can be easily embedded into any Java project or application deployed from a J2EE Web or application server.

Today consumers, employees, and corporations expect to receive invoices, account summary statements, and various online reports as part of their Web transactions. Reporting has become an essential requirement of every application or project as businesses, employees, and consumers continue to leverage Internet applications to complete transactions and manage business processes. This horizontal need for reporting functionality has increased the need for Java developers to use J2EE tools to quickly develop Web applications that can leverage the data access, security, and scalability of their existing Java application platforms.
www.reportingengines.com

PALM CHOOSES WEBSPHERE FOR HANDHELDS

(San Francisco) – Building on the relationship formalized last July with IBM, Palm, Inc., has licensed IBM's WebSphere Micro Environment (WME) Java 2 Micro Edition (J2ME) certified runtime environment for inclusion in future Palm Tungsten handhelds.

Uniting leaders in handheld computing and Java development, this agreement illustrates the strengthening relationship between Palm Solutions Group and IBM, and attests to the growing importance of embedded Java solutions in an era of pervasive computing. Together, Palm Solutions Group and IBM will offer an open standards–based and cost-effective development environment for mobile computing.

Palm Solutions Group chose IBM WME because it delivers the most powerful and flexible J2ME runtime environment available today and is compliant with the latest standards defined by the Java Community Process. This is the first time Palm Solutions Group has licensed a Java Virtual Machine for comprehensive distribution with its handhelds.
www.palm.com

IWAY ADAPTERS ALLOW CUSTOMERS TO REDUCE CUSTOM PROGRAMMING

(New York) – iWay Software, an Information Builders company and the world's leading supplier of business integration and adapter software, has announced expanded support for IBM's WebSphere product line, including WebSphere Application Server and the former MQSeries products, WebSphere MQ and WebSphere MQ Integrator (WMQI). iWay's Universal Adapter Framework will dramatically reduce the cost, time, and effort of integrating enterprise applications with WebSphere Application Server by dramatically reducing the need for custom programming.

Tight integration with WebSphere Studio, the platform's development environment, enables customers to easily create connections to mission-critical legacy data and packaged composite applications, according to iWay.
www.informationbuilders.com

DEBUGGING TUTORIAL

Take advantage of WebSphere Studio's powerful features

Remote Debugging

BY SANKAR VENKITACHALAM

This article will introduce you to the powerful remote debugging features of WebSphere Studio Application Developer 5.0 and show you how to debug a J2EE application deployed in a remote application server.

ABOUT THE AUTHOR
Sankar Venkitachalam is a senior developer at Bank One, where he focuses on designing, implementing, and supporting Bank One Card Services J2EE and content management projects. He has worked in J2EE-related technologies for the past three years and is certified in J2EE. Sankar holds a master's degree in computer science from Cleveland State University.

E-MAIL
deepsa_52@hotmail.com

I will walk you through the process of creating a simple Web application in WebSphere Studio, deploying it in a remote WebLogic server, and using the remote debugging features to debug Web application components such as a servlet and a JavaBean. I assume that you are already familiar with deploying a Web application on an application server, so I will not go into the details of Web application deployment.

Configuring the WebLogic Server for Remote Debugging

Sun introduced the Java Platform Debugging Architecture (JPDA) as a standard for debugging distributed Java applications. This standard allows tool and server vendors that support JPDA to seamlessly allow debugging on different host operating systems. In order to set up a WebLogic server for debugging, I added the following entry to the line in the WebLogic startup script that invokes the server:

```
-Xdebug -Xnoagent
-Xrunjdwp:transport=dt_sock-
et,server=y,address=60000,
suspend=n
-Djava.compiler=NONE
```

The following is a brief explanation of the debug parameters:

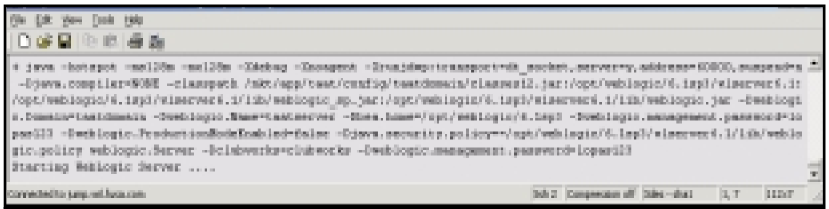
- ***-Xdebug:*** Enables debugging.
- ***-Xnoagent:*** Sun's classic VM supports both the old sun.tools.debug interface and JPDA; the -Xdebug option enables both, but defaults to running the sun.tools.debug agent. The -Xno-agent option turns this off so the JPDA will work. The HotSpot VM does not have this option. If you are using the HotSpot VM, this entry is optional.
- ***-Djava.compiler=NONE:*** Disables the JIT compiler. Debugging with the classic VM requires that its JIT compiler be disabled. For the HotSpot VM, this entry is optional.
- ***-Xrunjdwp:*** Loads in-process debugging libraries and specifies the kind of connection to be made. The transport=dt_socket option tells the debugger which transport mechanism to use. Specifying the dt_socket option allows the debugger to listen to a socket for incoming client connections. The other possible value is dt_shmem, which is applicable when both the debugger and the application server are on the same host. The server=y option indicates that the JVM is running in debug mode. The address= 60000 entry indicates that the server listens to incoming connections in port 60000. This has to be a free port in the system. The suspend=n option instructs the server not to wait until a debugger connection is established. If the suspend=y option is used, then the application server will suspend at the beginning of execution and wait until the debugger attaches to it.

Consult your application server documentation for details on the debugging parameters.

Figure 1 shows the output of my WebLogic 6.1 Server startup script, hosted on a Sun SPARC machine and listening to HTTP port 60000.

Creating a New Web Project

I updated my WebSphere Studio installation with the WebSphere Studio Application Developer fix pack PTF 001 (www3.software.ibm.com/ibmdl/pub/ software/websphere/studiotools/html/ptf001/wssd50/ install.html), which contains a number of



```
# java -hotspot -ms128m -mx128m -Xdebug -Xnoagent -Xrunjdwp:transport=dt_socket,server=y,address=60000,suspend=n
 -Djava.compiler=NONE -classpath /mkt/app/test/config/testdomain/classes12.jar:/opt/weblogic/6.1sp1/wlserver6.1:
/opt/weblogic/6.1sp1/wlserver6.1/lib/weblogic_sp.jar:/opt/weblogic/6.1sp1/wlserver6.1/lib/weblogic.jar -Dweblogi
c.Domain=testdomain -Dweblogic.Name=testserver -Dbea.home=/opt/weblogic/6.1sp1 -Dweblogic.management.password=lo
par123 -Dweblogic.ProductionModeEnabled=false -Djava.security.policy==/opt/weblogic/6.1sp1/wlserver6.1/lib/weblo
gic.policy weblogic.Server -Dclebworks=clebworks -Dweblogic.management.password=lopar123
Starting Weblogic Server .....
```

FIG. 1: OUTPUT OF WEBLOGIC SERVER STARTUP SCRIPT

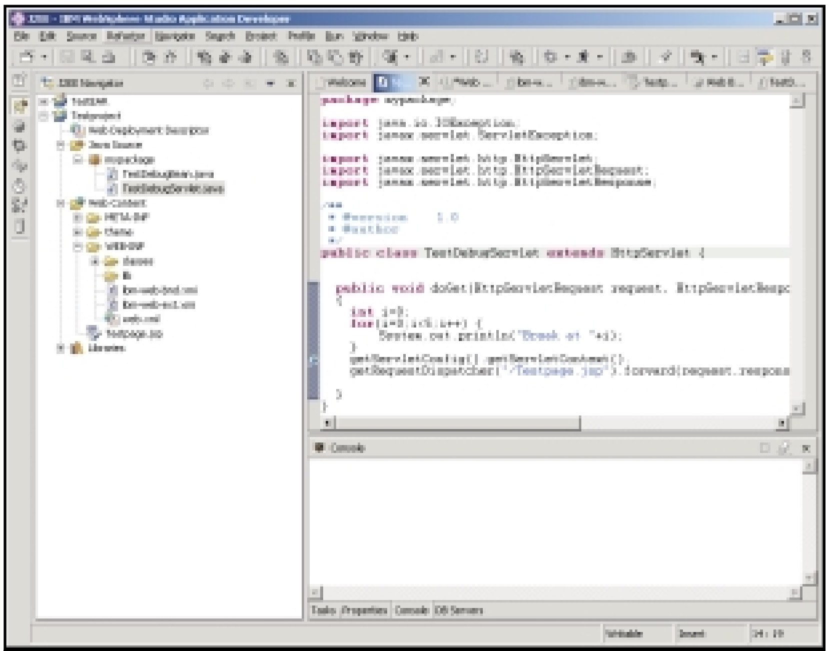
bug fixes. In order to create a new Web project, launch WebSphere Studio and do the following:

1. Click on "File", select "New", and "Web Project". Create a project named Testproject. Make sure you have J2EE 1.3 as the selected J2EE level. Create a new project under your workspace.
2. Add a new servlet, select the J2EE perspective on the left side of the IDE, right-click on Java Source under the Testproject, and select "New" followed by "Servlet". Follow the prompts to create a servlet named TestDebugServlet in "mypackage". The source code for TestDebugServlet is shown in Listing 1.
3. Add a new JavaBean, select the J2EE perspective, right-click on Java Source under the Testproject, and select "New" followed by "Class". Follow the prompts to create a JavaBean named TestDebugBean in a package "mypackage". The source code for TestDebugBean is shown in Listing 2.
4. Add a new JSP page, select the J2EE perspective, right-click on "Web Content" under the Testproject, and select "New" followed by "JSP". Follow the prompts to create a JSP page named Testpage.jsp. The source code for Testpage.jsp is shown in Listing 3.
5. Modify the web.xml under the WEB-INF folder, as shown in Listing 4. I have /TestDebugServlet as the URL pattern for TestDebugServlet.

Once you have completed creating the Web application, you will have a view similar to that shown in Figure 2.

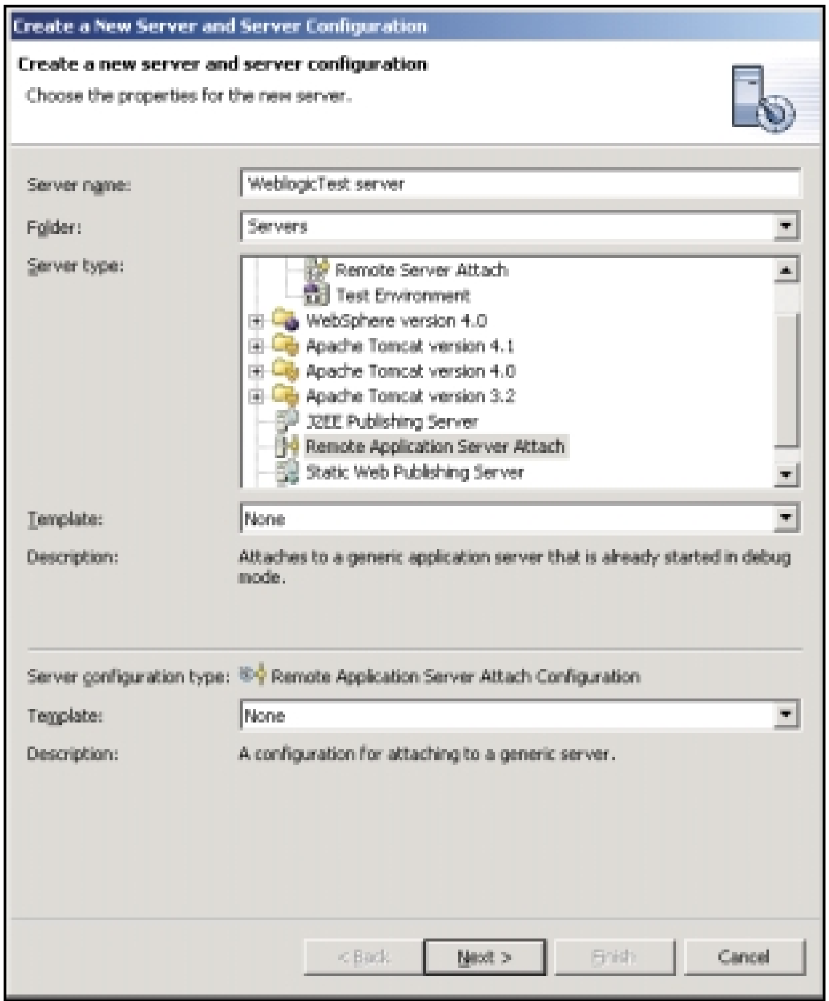
Deploy the Web Application

The project files, namely the JSP file, the web.xml file, and the class files, will be located under the project workspace. I used the context name "Testproject" for the Web application. It is very handy to have a simple FTP script to automate the job of transferring the files to the appropriate location in the remote application server installation. If you are using Ant to build your application, you may use the FTP Ant task to automate file transfer. Refer to [http://ant.apache. org/ manual/OptionalTasks/ftp.html](http://ant.apache.org/manual/OptionalTasks/ftp.html) for



```
package mypackage;

import java.io.IOException;
import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * @version    1.0
 * @author
 */
public class TestDebugServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletRespo
  {
    int i=0;
    for(i=0;i<5;i++) {
        System.out.println("Break at "+i);
    }
    getServletConfig().getServletContext().
    getRequestDispatcher("/Testpage.jsp").forward(request,respons

  }
}
```

FIG. 2: THE J2EE PERSPECTIVE



Create a New Server and Server Configuration

Create a new server and server configuration

Choose the properties for the new server.

Server name: WeblogicTest server

Folder: Servers

Server type:
- Remote Server Attach
- Test Environment
- WebSphere version 4.0
- Apache Tomcat version 4.1
- Apache Tomcat version 4.0
- Apache Tomcat version 3.2
- J2EE Publishing Server
- Remote Application Server Attach
- Static Web Publishing Server

Template: None

Description: Attaches to a generic application server that is already started in debug mode.

Server configuration type: Remote Application Server Attach Configuration

Template: None

Description: A configuration for attaching to a generic server.

FIG. 3: CREATE A NEW SERVER AND SERVER CONFIGURATION DIALOG BOX

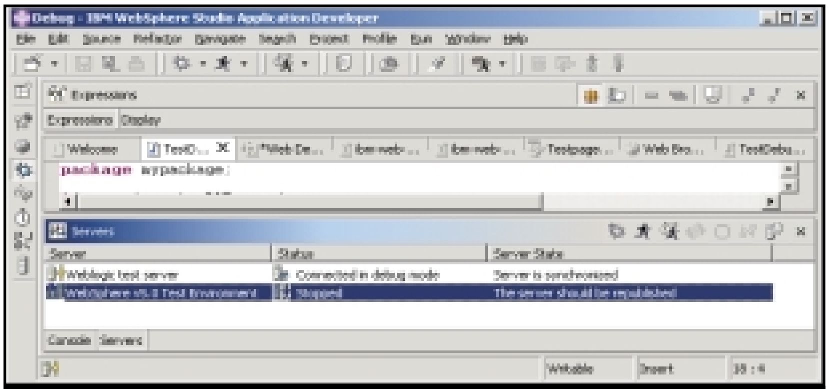
details. During development, you may choose to use either a WAR file deployment or an exploded Web application directory structure. Once the Web application is compiled and built, the process of transferring the files to the remote server should be no more than a mouse-click away.

Create a Server Configuration for Remote Debugging

Follow the steps below for creating a new server configuration for the remote WebLogic Server.

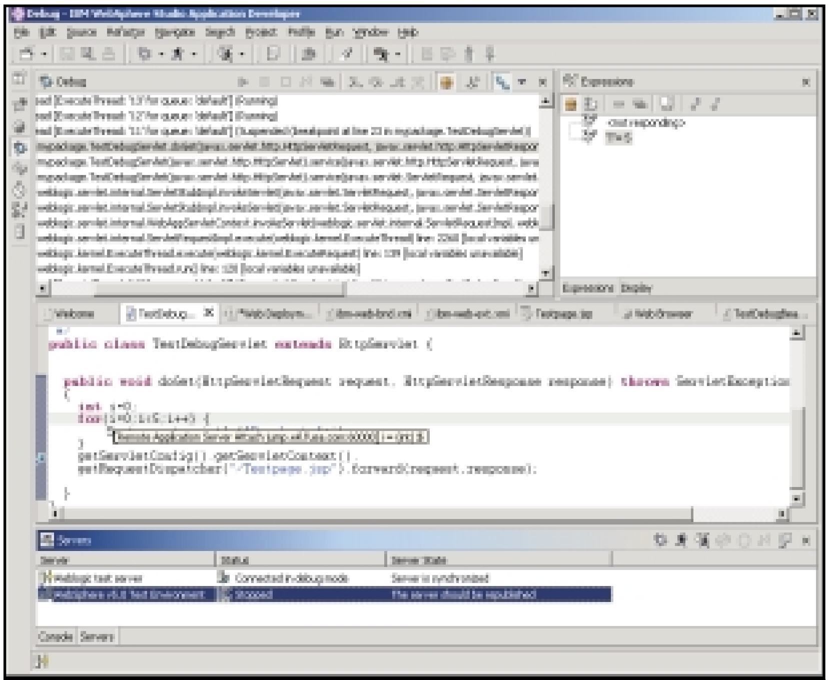
1. On the top menu bar, click on "Window", "Show view", and select "Server Configuration". You may have to select the "Server Configuration" option from the "Other.." menu item, depending on whether or not you have made a previous selection.
2. On the server configuration pane, right-click on "Servers", click on "New", "Server and Server Configuration". You will be prompted with a dialog box, as shown in Figure 3.
3. Select "Remote application server attach" from the list and enter a server name, for example, "WebLogic test server". You will be prompted to enter the hostname, JVM debug port, and HTTP port. For the Testproject, my values are 6000 for the HTTP port and 60000 for the debug port.

Now you have configured a server that can be used for remote debugging.



Server	Status	Server State
Weblogic test server	Connected in debug mode	Server is synchronized
WebSphere v5.0 Test Environment	Stopped	The server should be republished

FIG. 4: THE SERVERS PANE



```
public class TestDebugServlet extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException
  {
    int i=0;
    for(i=0;i<5;i++) {
    }
    getServletConfig().getServletContext().
    getRequestDispatcher("/Testpage.jsp").forward(request,response);
  }
```

Server	Status	Server State
Weblogic test server	Connected in debug mode	Server is synchronized
WebSphere v5.0 Test Environment	Stopped	The server should be republished

FIG. 5: THE DEBUG PANE

Start Remote Debugging

Debugging the application is as simple as connecting the server you have configured to your application server and typing in the URL of the Web application on a browser. Follow the following steps to attach the WebSphere debugger to the remote application server:

1. On the top menu bar, click on "Window", "Show view", and select "Server". You may have to select the "Server" option from the "Other.." menu item, depending on whether or not you have made a previous selection.
2. Right-click on the "WebLogic test server" that you created earlier, and click on debug. If your application server is configured properly, your debugger will connect to it. A "Servers" pane similar to that shown in Figure 4 will appear.

Now you can switch to the Debug perspective, the one with the bug symbol on the left side. You can set breakpoints on the servlet or on the JavaBean by double-clicking on the left side of the source code pane. Once you launch the browser to point to the TestDebugServlet on your application server, WebSphere Studio will intercept your calls and you can break at the location that you desire.

Note: You may have to select "Window", "Show view", and "Debug" again to view the Debug pane. Now you can see the various debug options, as shown in Figure 5. You can also watch variable values by selecting the variable of interest in the code window and viewing the value on a tool tip.

Conclusion

WebSphere Studio 5.0 comes with powerful debugging features that allow you to debug applications on local machines as well as remote application server hosts. Once the code, build, deploy, and debug cycle has been set up properly for a project, debugging an application on a different JVM on a different host operating system should be a smooth task.

LISTING 1: TESTDEBUGSERVLET.JAVA

```
package mypackage;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestDebugServlet extends HttpServlet {

  public void doGet(HttpServletRequest request,
         HttpServletResponse response) throws
ServletException,                      IOException  {
    int i=0;
    for(i=0;i<5;i++) {
        System.out.println("Break at "+i);
    }
    getServletConfig().getServletContext().
    getRequestDispatcher("/Testpage.jsp").forward(request,
      response);

  }
}
```

LISTING 2: TESTDEBUGBEAN.JAVA

```
package mypackage;
import java.util.Date;
public class TestDebugBean {

  public TestDebugBean() {
  }

  public Date getDate() {
    Date d=new Date();
    String datestring=d.toString();
    return d;
  }
}
```

LISTING 3: TESTPAGE.JSP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<%@ page language="java" contentType="text/html"%>

<jsp:useBean id="mybean" scope="page"
class="mypackage.TestDebugBean"></jsp:useBean>
<html>
<head>
<title>
Hello Remote Debugger
</title>
</head>
<body>
<h2>
The current time is: <%= mybean.getDate() %>
</h2>
</body>
</html>
```

LISTING 4: WEB.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
          Application 2.3//EN" "http://java.sun.com/dtd/web-
                    app_2_3.dtd">
<web-app id="WebApp">
   <display-name>Testproject</display-name>
   <servlet>
      <servlet-name>TestDebugServlet</servlet-name>
      <display-name>TestDebugServlet</display-name>
      <servlet-class>mypackage.TestDebugServlet</servlet-
         class>
   </servlet>
      <servlet-mapping>
         <servlet-name>TestDebugServlet</servlet-name>
         <url-pattern>/TestDebugServlet</url-pattern>
      </servlet-mapping>
</web-app>
```

WSAD/ORACLE TUTORIAL

-continued from page 38

```
  public int getDebug() {

         return (this.debug);

  }

  /**
  * Retreive the data from its persistent storage.
  *
  * @exception Exception if any problem occurs while loading
  */
  private synchronized void retrieve() throws Exception {

         // First create the local vars for the query
         DataSource ds = null;
         Connection conn = null;
         PreparedStatement statement = null;
         ResultSet rs = null;
         InitialContext ic = null;
         // Put your select statement here !!
         String queryString = ("SELECT * FROM MYTABLE");
         // Put your column name here!!
         String columnName = ("DNAME");
         String columnValue = null;

         try {
                   // The Initial Context
                   ic = new InitialContext();
                   // The Datasource object
                   ds = (DataSource) ic.lookup("jdbc/ds1");
                   // Obtain a Connection from the
                   // DataSource using userid and password
                   conn = ds.getConnection("was","was");

         } catch (NamingException lx) {
                   System.out.println("error in lookup: " +
                   lx.getMessage());
         } catch (SQLException ex) {
                   System.out.println("error during connec-
                   tion: " + ex.getMessage());
         }

         try {
                   // Create the statement passing the
                   // querystring
                   statement =
                             conn.prepareStatement(
                                       queryString,
                                       rs.TYPE_SCROLL_INSEN-
                                       SITIVE,
                                       rs.CONCUR_READ_ONLY);
                   // execute the query
                   rs = statement.executeQuery();

                   // See if any data was returned
                   boolean test = rs.first();


// Loop through all data and output the results to the
// console
                   while (test) {
                   // MODIFY THIS VALUE TO MATCH THE COLUMN
                   // OF YOUR TABLE !!!!
                             columnValue =
                             rs.getString(columnName);

                             System.out.println("Database
                             result: " + columnValue);

                             // Get the next row
                             test = rs.next();

                   }
         } catch (SQLException ex) {
                   System.out.println("error during
                   execute");
         }

  }

}
```

A Conversation with IBM's Kerrie Holley

Chief architect of e-Business Integration Solutions discusses e-business on demand

As chief architect of e-Business Integration Solutions for IBM Global Services, Kerrie Holley turns business requirements into cutting-edge network solutions. Holley, an IBM Distinguished Engineer, was honored for his contributions to IBM when he received the Chairman's Award at the 2003 Black Engineer of the Year Awards ceremonies.

In an exclusive interview with *WebSphere Developer's Journal* editor-in-chief Jack Martin, Holley discusses the origins and future of e-business on demand, and the importance of a service-oriented architecture in today's enterprise.

WSDJ: When you first started at IBM what did you go there to do?
Holley: I have always been part of IBM's services organization, so when I first joined IBM in 1986, I joined what was then an independent business unit, which was a predecessor – a very early predecessor – to what is now the services business in IBM.

WSDJ: It sounds as if you were one of Global Services' original employees?
Holley: Exactly.

WSDJ: How did you get involved in WebSphere and Web services?
Holley: It's a natural growth. As an architect within Global Services, I tend to focus predominantly on emerging trends and emerging technologies, and clearly Web services on demand is just that, so doing work in WebSphere – doing work in the whole J2EE space – is a large part of what we do in terms of helping customers meet the vision of Web services on demand. It is a natural outgrowth from a lot of the e-business development work we were doing just a few years back.

WSDJ: What is the difference between Web services on demand and e-business on demand?
Holley: A very good question. Web services on demand – a term that we actually haven't often heard used – is a subset of e-business on demand. That is, Web services on demand would be an enabler for the full vision of e-business on demand.

WSDJ: What types of customers are interested in instituting Web services strategies?
Holley: It's actually cross-industry. I work mainly in the financial marketplace where you see a lot of credit card companies, banks, brokerage firms – and it's especially true there because they have a lot of external markets and customers, like a credit card company wants to reach member banks. Brokerage houses want to talk to other institutions and to other customers, so they have a lot of B2B scenarios, which plays nicely with Web services.

WSDJ: What are the top three B2B scenarios that you see customers using?
Holley: One is simple integration with an external supplier. That's simply the ability to use open standards in a way that allows them to talk without having point-to-point solutions, without having to have the two IT departments go through some coordinated development activity to bring a solution or product to market. So one scenario is simple integration. Another scenario is enabling a partner to have seamless access to a service that you may already have, that traditionally you provide internally, but now you want to make externally available.

Let's take a credit card company, where you know the typical problem you have. When you or I have a dispute about our credit card, we call up the credit card company and we deal with their customer service organization and in turn the customer service organizations of the credit card company and the retailer sort of talk, resolve, and get back to us. In a future scenario I would be able to go into that credit card company's back-end system through a type of Web service and extract the information I want so it would help me with the dispute process.

A third scenario is probably what I see most often in internal integration, where you have disparate computing platforms and you need to find a seamless way to make the systems talk to each other. That's an internal integration issue within the enterprise, but it's a pretty typical example of what organizations start with, especially where they have grown through mergers and acquisitions and that's been a part of their business model. They integrate the silos that have been built into the company into a more horizontal scheme.

WSDJ: I think the concept of Web services on demand is probably new to a lot of our readers. Could you explain first how they play with e-business on demand; second, exactly what Web services on demand is; and third, where you see that playing out in the marketplace early on?

Holley: I think that when we talk about Web services we must also talk about another concept that is really tightly coupled with Web services – service-oriented architectures. It is through the combination of Web services and service-oriented architecture that we really have this Web services on demand.

WSDJ: Can you explain what service-oriented architecture is?

Holley: Service-oriented architecture is where you have an architecture that is predominantly composed of services. You have a consumer of those services, you have a provider of those services, you have a way of connecting the two together – some kind of matchmaking capability between the two, and more important, this service has a predictable result when you invoke it. It contrasts with what we typically do, where we may have different types of results coming back from the traditional application programming interfaces.

Service-oriented architecture allows the design of software systems that provide services to other applications through published and discoverable interfaces where the services can be invoked over a network. When customers implement service-oriented architectures using Web services technologies we create a more powerful and flexible programming model for building software. Development and ownership costs are reduced, as well as implementation risks.

WSDJ: Back now to the original question on e-business on demand and how it all plays out.

Holley: Web services really means two things. It means Web services technologies – SOAP, UDDI, WSDL and XML, and so on. But it also means the service itself, the encapsulated business function that we want to tap and have access to and make available. When we talk about Web services we are really using it in two contexts. We are talking about it from a technology vantage point and from a service standpoint. That's why I say you also have to have the service-oriented architecture (SOA). When you put those two together, you have something very powerful – Web services on demand – because now you have the ability to do this assembly of services, this plug-and-play type of software development model, this nirvana that we all want to achieve.

What this provides for the business is, obviously, speed to market; they get flexibility because they can leverage the services and mix and match them in a way that makes sense for their business; they get technology agnostic to some degree in that they are no longer tied to a specific platform; it doesn't matter if the service is implemented on technology A or B. Obviously we have technology like WebSphere that facilitates this, but another benefit of a technology-agnostic approach is the heterogeneity that you can provide in this kind of solution so customers can pick and choose what makes sense for their business and optimize their total cost of ownership.

WSDJ: Are you saying that your Web services on demand will work with .NET?

Holley: Well, in Web services on demand definitely, the answer is yes. It is technology agnostic, so it works with – and allows customers to pick and choose – the technology that makes sense for their business

WSDJ: How does this all play with e-business on demand?

Holley: e-business on demand is a much broader initiative. e-business on demand deals with the application environment, which is a lot of what I focused our attention on just now. But e-business on demand also deals with the operating environment and its infrastructure aspects. It's about autonomic computing, so that you have all of the attributes that give you lights-out, 24x7, this self-healing, self-optimizing, and self-configuring environment that is another aspect of on demand. Another aspect of on demand is the whole ability to leverage IT as a utility. That's another aspect of on demand.

Then you have the business aspect of on demand as well, where suddenly, because I am leveraging all these technologies I can become this on-demand business that becomes more resilient, more variable, and more adaptive to market changes. Web services on demand really becomes an enabler of that overall vision, hence the two concepts are definitely linked but different. Web services on demand is a subset, an enabler – and the technology is enabling technology to make that happen.

WSDJ: It sounds like the Web services on demand play is the software side of the engine to make e-business on demand a reality.
Holley: Exactly. It helps in a couple of dimensions. It helps on the application side that we've talked about, in the context that we can move to this plug-in, assembly-type model of building solutions, but also on the infrastructure side it helps us implement some of the solutions because we are using those Web services technologies as the open standard to substantiate some solutions in that space.

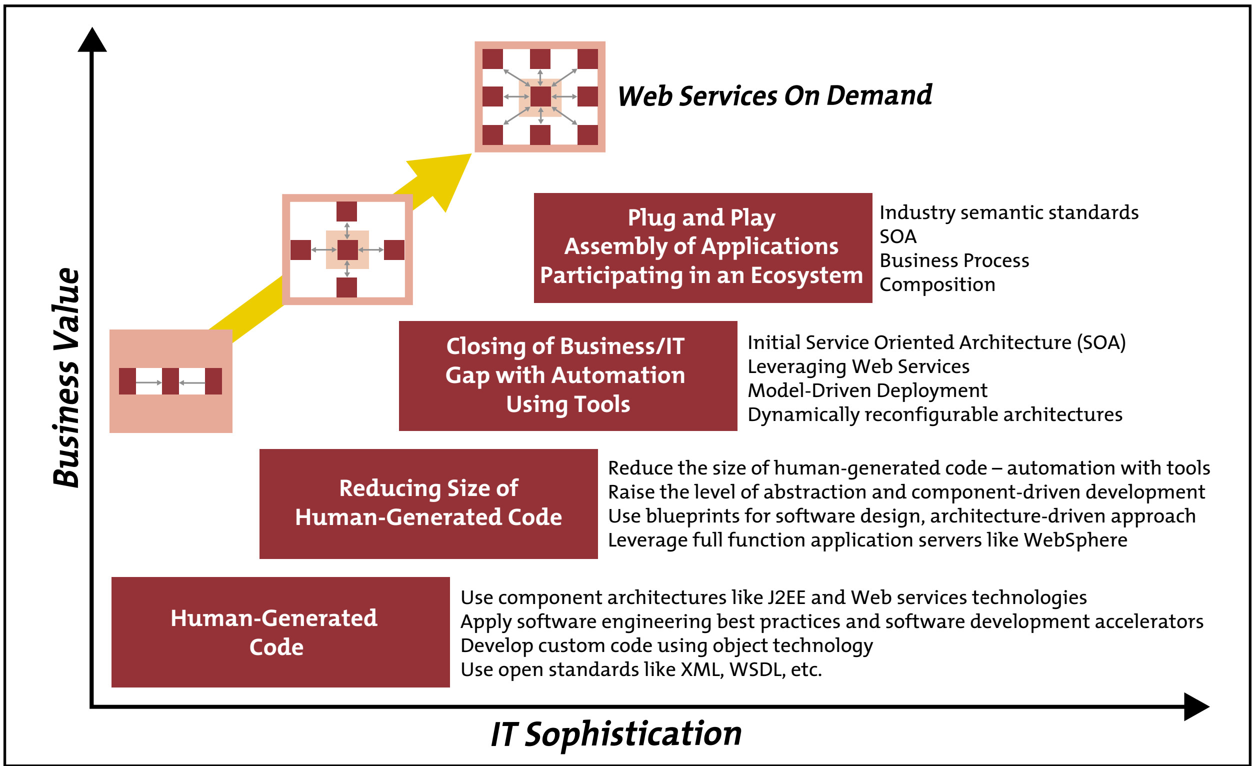
WSDJ: Can you see this driving down costs?
Holley: Definitely. Obviously that is going to take place over time. What I've typically seen is that whenever you talk about driving costs down, the next question is whether you can quantify it. Can you establish what that cost is going to be? That becomes more difficult because you are driving down cost through efficiency and a lot of dimensions, but that efficiency is eaten up with other demands of the business.

WSDJ: Your early customers, how are they using that today?
Holley: Our early customers are focused on some of the attributes of responsiveness and variableness. Those are probably two of the key attributes that drive their interest in these models. When they build solutions they don't want those solutions to be legacy solutions tomorrow; they want those solutions to rapidly accommodate and adjust to the changing business landscape. I see that a lot in terms of trying to have integrated systems, speed to market, more variable costs, and a better return on investment.

WSDJ: How do the new features in WebSphere 5 help you do this?
Holley: We have improved support for Web services, and that's probably the biggest factor, so as a technology it is a basic building block and it provides the broadest Web services platform support available. This includes the ability to generate Web services using wizards, service choreography of end-to-end business processes, reliability through JMS and message-driven bean support, security using the Web services gateway, and interoperability using the Web Services Invocation Framework.

WSDJ: A lot of our readers are technologists and developers. They would be very interested, I think, in knowing what basic components they need to have present to try their own Web services on demand play.
Holley: I think you want to lead with architecture. Establishing an architecture is key because the architecture lays out the basic framework and the basic components, and enables you to select the appropriate technology. I know I mentioned the opportunity with the service-oriented architecture, but that is one part of it. Let's take it bottom up for a minute. If we look at a maturity model that organizations would evolve to as they look to reach Web services on demand, there is a lot of technology that has been out on the market for some time. Object technology is one, and it will be a basic building block here. Another basic building block is going to be component technology, which you get from things like J2EE.

We started with object technology, and then saw more advances to things like component models. With component models we are also getting things like frameworks that we provide, and some of those are open source. In both cases, those become basic building blocks for an organization looking to


Web Services On Demand
Plug and Play Assembly of Applications Participating in an Ecosystem
Industry semantic standards
SOA
Business Process Composition
Closing of Business/IT Gap with Automation Using Tools
Initial Service Oriented Architecture (SOA)
Leveraging Web Services
Model-Driven Deployment
Dynamically reconfigurable architectures
Reducing Size of Human-Generated Code
Reduce the size of human-generated code – automation with tools
Raise the level of abstraction and component-driven development
Use blueprints for software design, architecture-driven approach
Leverage full function application servers like WebSphere
Human-Generated Code
Use component architectures like J2EE and Web services technologies
Apply software engineering best practices and software development accelerators
Develop custom code using object technology
Use open standards like XML, WSDL, etc.
Business Value
IT Sophistication
FIG. 1: WEB SERVICES ON DEMAND ADOPTION MODEL

establish this. Then, obviously, picking the right technologies – technologies like WebSphere – that help you instantiate specific solutions. And above all, if you look at the service-oriented architecture, or even below that, you've got architectures that are organized around layers, with which there is a separation of concerns. These layers allow you to organize teams around those layers, which allows you to build parts of your solution by layer. This in turn promotes multisite development. I can have 100 people or 1,000 people in India, or 1,000 people someplace else; this provides an enterprise with variableness in their deployment of resources.

I envision organizations moving toward this maturity model by starting with objects, exploiting frameworks, moving to component development, applying software accelerators – but ultimately everything being part of a holistic approach to create the service-oriented architecture with services that in many cases can be assembled. For example, the list balance inquiry or open account service will be available to the teller application, the personal banking application, the telephony solution, the mortgage banking application, or other applications external to the enterprise. We effectively reduce the silos and achieve integration using Web services.

The realization of service-oriented architectures will allow organizations to build solutions faster to develop software faster. Of course, the fastest way to develop software is not to write any software but to assemble software with a plug-and-play model much like kids assemble structures using toy blocks. Organizational maturity to realize this state occurs by reducing the amount of human-generated code, raising the level of abstraction, using model-driven development tools like those provided with Rational software, and exploiting technologies like WebSphere. Figure 1 illustrates this adoption model as organizations realize Web services on demand.

WSDJ: Which industries are showing interest in getting involved with Web services on demand right now?

Holley: Definitely the financial and banking markets are; manufacturing is another one; and insurance – which is still in the financial market, but is an industry that we see moving quickly in this space – and automotive as well.

WSDJ: So you are pulling a Web services play and let's say there is some business logic and at the end of the day they sell the customer something to push it to commerce or whatever. Do you see what I want to try to get from you?

Holley: Web services really is the basic building block that makes this possible and ties it all together with WebSphere. If we look not only at the capabilities that we have today out of version 5, but look down the road, we have the ability to eventually make this technology have either a small or large footprint. That's the variable part: a small footprint for a one-person development shop or a larger footprint for the hundreds of developers in a complex enterprise. A second benefit is the fact that we can close this business-to-IT gap so that we can actually begin to do modeling and be able to instantiate those models with the technology that we get from the WebSphere family combined with IBM's Rational software suite. From an integration standpoint, I am getting a lot of integration with WebSphere, as well as with an open standards base that's reliable, secure – and that's another big benefit that I am getting out of that product suite.

WSDJ: If a WebSphere customer reads this interview and says, "Okay, I want to try this Web services on demand concept," what would be an easy area in which to implement Web services on demand today?

Holley: That would be a simple within-the-enterprise integration opportunity where you have a real business need to do some integration to provide some value to your customer base. That would be a significant opportunity that would give you some significant business value, but you would also be smart by starting small. The technology is proven, your risks are small, and your opportunity for success is great.

WebSphere DEVELOPER'S JOURNAL

Advertiser Index...

COMPANY	URL	PHONE	PG
DIRIG SOFTWARE	WWW.EBIZPERFORM.COM	603-889-2777	21
GLOBAL KNOWLEDGE	WWW.GLOBALKNOWLEDGE.COM	800-COURSES	5
KENETIKS	WWW.KENETIKS.COM	888-KENETIKS	13
LINUXWORLD CONFERENCE & EXPO	WWW.LINUXWORLDEXPO.COM	800-657-1474	28-29
MACROMEDIA	WWW.MACROMEDIA.COM/GO/WSDJ/	415-252-2000	51
MQSOFTWARE	WWW.MQSOFTWARE.COM	952-345-8720	6-7
PROLIFICS	WWW.PROLIFICS.COM/WEBSERVICES	800-675-5419	2
QUEST SOFTWARE	HTTP://JAVA.QUEST.COM/QCJ/WDJ	800-663-4723	19
SYS-CON PUBLICATIONS	WWW.SYS-CON.COM/2001/SUB.CFM	888-303-5282	35
TRILOG GROUP	WWW.FLOWBUILDER.COM	800-818-2199	52
VERITAS	WWW.VERITAS.COM	650-527-8000	11
VERSATA	WWW.VERSATA.COM/BUSINESSLOGICDESIGNER	800-984-7638	23
WEB SERVICES EDGE WEST 2003	WWW.SYS-CON.COM/WEB SERVICES2003WEST	201-802-3069	48-49
WILY TECHNOLOGY	WWW.WILYTECH.COM	888-GET-WILY	3

WebSphere DEVELOPER'S JOURNAL

Coming Next Month...

CASE STUDY

The ServiceLocator Design Pattern:
A Template for Streamlining Future Development

BY LLOYD HAGEMO AND RAVI KALIDINDI

INTEGRATION

Enabling WebSphere Studio Application Developer Integration Edition V4.11 JCA Applications to Run in WebSphere Application Server Advanced Edition V4.04

BY SANDY MINOCHA

EJB TUTORIAL

Step-By-Step EJB 2.0 Inheritance

BY ASHIM RANJITKAR, RAJU MUKHERJEE, AND SOUTIK SINGHA

WSAD

Profiling: Using WSAD 5.0's
Full-Featured Toolset

BY ANDREW SONDGEROTH

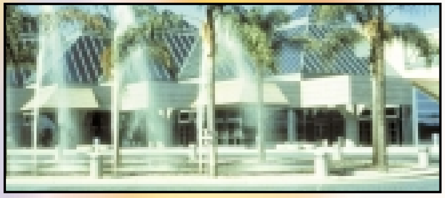
International Web Services Conference & Expo

Web Services Edge WEST 2003

web services EDGE conference&expo

SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



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

Featured technologies and topics will include:

- Focus on Java
- Focus on Web Services
- Focus on Mac OS X
- Focus on XML
- Focus on .NET

XML

WebSphere

Microsoft .net

Mac OS X

COMING IN 2004

web services EDGE conference&expo

BOSTON

FEBRUARY 24-26, 2004

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.

CONTACTS:

Exhibit & Sponsorship	*Conference & Education*
GRISHA DAVIDA	MICHAEL LYNCH
201 802-3004	201 802-3055
grisha@sys-con.com	mike@sys-con.com

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

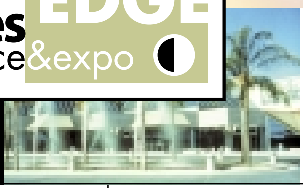
Web Services Edge WEST 2003

WEB SERVICES EDGE CONFERENCE & EDUCATION

web services EDGE conference&expo

SEPT. 30 - OCT. 2, 2003

Santa Clara, CA



JAVA TECHNOLOGY

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

Sessions will focus on:

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

.NET TECHNOLOGY

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

Sessions will focus on:

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

WEB SERVICES TECHNOLOGY

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

Sessions will focus on:

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

MAC OS X

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

Sessions will focus on:

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

XML TECHNOLOGY

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

Sessions will focus on:

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

MEDIA SPONSORS:

WebServices Journal · JAVA Developer's Journal · WebSphere Developer's Journal · XML Journal · .NET Developer's Journal · WebLogic Developer's Journal · LinuxWorld Magazine · wireless Business & Technology · Linux Business Week · CF Advisor · ColdFusion Developer's Journal · PowerBuilder Developer's Journal · market WIRE · HSP Street.com · OASIS · SD Times · LinuxWorld.com · SAMS · realindustry

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

PRODUCED BY SYS-CON EVENTS

Using WebSphere Portal for Your Intranet Framework

BY ROGER BERGER

In the past, corporate intranets were built from the ground up with various tools and technologies. Some companies hoped to establish a solution that would not only address a current problem but be flexible enough to handle future expansion. Unfortunately, building a flexible infrastructure for your own intranet while addressing the business needs of users is difficult to do. Fortunately, IBM has developed the WebSphere Portal product, which offers a framework that allows you to focus on content and business issues rather than structural and navigational issues. Companies can leverage this established, but flexible, portal framework for administration, searching, content management, and application integration.

So why use a portal? Portals unify a wide range of information, projecting a comprehensive view for your users, thereby acting as a single entry point for all information. This allows users to do their jobs more effectively, increases productivity, and improves timely access to data. Homegrown intranets typically patch together various pieces of functionality over time; a portal framework will allow for a consistent look and feel to various applications within the environment, along with a unified security model.

Portals should be built to make users' lives easier. Successful portal solutions will ensure that the end users have bought into the value of a portal and use it throughout their day. End users tend to focus on the importance of content and streamlined access over qualities such as style, look, and feel. Including user input in the introduction and piloting of a portal, and focusing on the following best practices will aid in acceptance of the new technology:

- Leverage a variety of search technologies within your portal to ensure easy access to data.
- Implement a suite of personalization services to help your organization get information to the right people at the right time.
- Leverage "single sign-on" technologies to simplify the authentication process for various resources within the environment.
- Integrate your internal and external applications; a portal framework should allow for straightforward integration of these assets in a manageable fashion.

IBM's WebSphere Portal, in conjunction with various Lotus collaborative applications, offers companies a fully functioning portal framework that incorporates best practices for usability, integration, enhanced collaborative functionality, and administration. WebSphere Portal incorporates a simplified look and feel by leveraging the use of places, pages, and portlets throughout the framework. Places represent the higher navigational elements of the portal, while pages represent the container for portlets containing value-added business data that users ultimately access. This consistent navigational strategy allows streamlined access to various types of data. There are numerous portlets that IBM offers (newsfeed, CRM, SFA, etc.) that can be downloaded easily and incorporated into a portal.

WebSphere Portal is based on J2EE standards, allowing easy integration of other J2EE and Web-based applications into the framework. Applications can integrate into the portal environment without modification. However, for those applications that need integration services, there are a variety of tools available – provided by IBM or purchased from other vendors – to take applications and "portletize" them to ensure functionality within the portal framework. Incorporation of Web services and LDAP-compliant applications can also be done within the portal framework.

The collaborative elements that Lotus offers are tightly integrated into the overall WebSphere Portal solution. Lotus Sametime, QuickPlace, Discovery Server, Dom.Doc, and Extended Search are ideal elements that can add significant value to your WebSphere Portal solution. This will allow you to add contextual collaboration and Sametime-enabled elements into your portal via the Lotus suite, leading to increased ease of access to data and human resources that drive ROI for your portal investment. The ability to incorporate these elements into your portal – along with the single sign-on, enhanced search, and personalization capabilities that are inherent in the product – is a significant reason to consider WebSphere Portal. While streamlining the user experience, IBM has also incorporated a straightforward common interface to the underlying administrative functions of WebSphere Portal. Administrators log in to WebSphere Portal with specific administrative rights granted through a role-based security model, and are presented with the tools necessary to maintain the environment.

My personal preference is to stay away from proprietary portal solutions; a company's best bet is to focus on a framework that leverages standards. IBM's J2EE-based WebSphere Portal incorporates best practices into its framework and integrates with other Web-based and J2EE applications, along with Lotus' best-of-breed application suite, ensuring a high value proposition for your portal investment.

ABOUT THE AUTHOR... Roger Berger is a director and cofounder of Rave Software Solutions, a division of Computer Generated Solutions (CGS), Inc. Roger has been instrumental in defining and executing CGS's strategies around its portal, collaboration, messaging, and wireless offerings. Along with hosting the IBM WebSphere Innovation Center in New York City and its Virtual IBM WebSphere Innovation Center, CGS delivers a broad range of solutions in the e-business, application development, learning, and help desk space.

E-MAIL
rberger@cgsinc.com

MACROMEDIA

WWW.MACROMEDIA.COM/GO/WSDJ/

TRILOG GROUP

WWW.FLOWBUILDER.COM