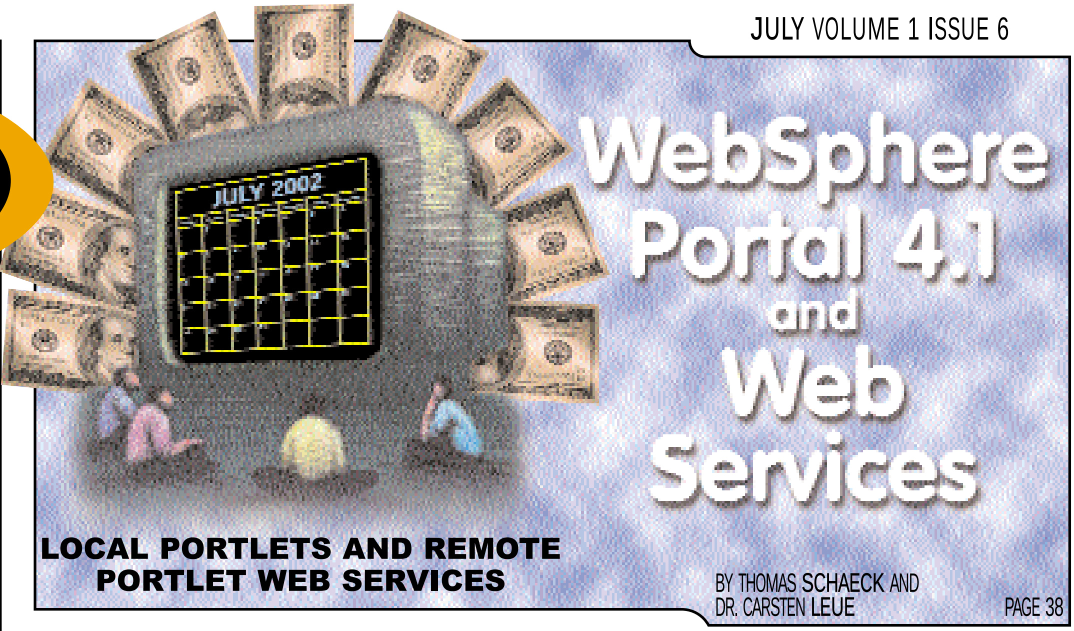
INTERVIEW: IBM's VICE PRESIDENT OF COMPETITIVE TECHNOLOGIES PAGE 46

WebSphere® DEVELOPER'S JOURNAL

The World's Leading Independent WebSphere Developer Resource

WebSphereDevelopersJournal.com

JULY VOLUME 1 ISSUE 6



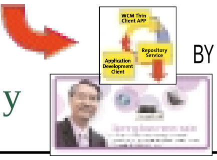
WebSphere Portal 4.1 and Web Services

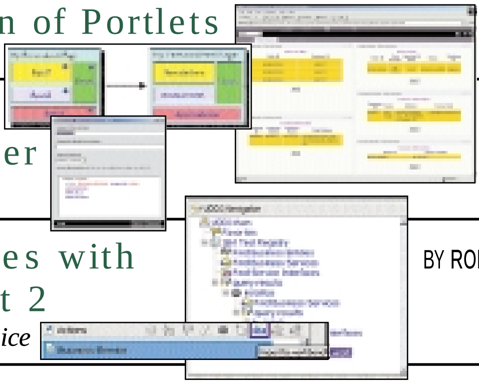
LOCAL PORTLETS AND REMOTE PORTLET WEB SERVICES

BY THOMAS SCHAECK AND DR. CARSTEN LEUE PAGE 38

DISPLAY UNTIL SEPTEMBER 30, 2002
$15.00US $16.00CAN
07>
0 71486 03422 3

SYS-CON MEDIA





ADVISORY BOARD
Richard Arone, James Martin, Greg Williams

Editor-in-Chief	JACK MARTIN
Editorial Director	JEREMY GEELAN
Executive Editor	GAIL SCHULTZ
Contibuting Editors	PATTI MARTIN JAMES MARTIN
Special Projects Editor	RICHARD ARONE
Managing Editor	CHERYL VAN SISE
Senior Editor	M’LOU PINKHAM
Editor	NANCY VALENTINE
Associate Editors	JAMIE MATUSOW JEAN CASSIDY
Assistant Editor	JENNIFER STILLEY

WRITERS IN THIS ISSUE
Ron Ben-Natan, Samar Choudhary, Joe Farsetta, Michael T. Ferro, Gary Goldberg, David B. Lection, Carsten Leue, Jack Martin, Cody Menard, Shankar Ramaswamy, Sai Rathnam, Amber Roy-Chowdhury, Thomas Schaeck, Robert Will, Douglass J. Wilson

SUBSCRIPTIONS
For subscriptions and requests for Bulk Orders,
please send your letters to Subscription Department.
SUBSCRIBE@SYS-CON.COM
Cover Price: $15/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

PUBLISHER, PRESIDENT AND CEO	FUAT A. KIRCAALI
COO/CFO	MARK HARABEDIAN
VP, BUSINESS DEVELOPMENT	GRISHA DAVIDA
SENIOR VP, SALES & MARKETING	CARMEN GONZALEZ
VP, PRODUCTION & DESIGN	JIM MORGAN
VP, SALES & MARKETING	MILES SILVERMAN
VP, EVENTS	CATHY WALTERS
ASSISTANT CONTROLLER	JUDITH CALNAN
ACCOUNTS PAYABLE	JOAN LAROSE
ACCOUNTS RECEIVABLE	JAN BRAIDECH
ACCOUNTING CLERK	BETTY WHITE
ADVERTISING ACCOUNT MANAGERS	MEGAN RING-MUSSA ROBYN FORMA
ASSOCIATE SALES MANAGERS	CARRIE GEBERT ALISA CATALANO KRISTIN KUHNLE LEAH HITTMAN
CONFERENCE MANAGER	MICHAEL LYNCH
REGIONAL SALES MANAGERS	MICHAEL PESICK RICHARD ANDERSON
ART DIRECTOR	ALEX BOTERO
ASSOCIATE ART DIRECTORS	AARATHI VENKATARAMAN LOUIS CUFFARI CATHRYN BURAK RICHARD SILVERBERG
ASSISTANT ART DIRECTOR	TAMI BEATTY
WEBMASTER	ROBERT DIAMOND
WEB DESIGNERS	STEPHEN KILMURRAY CHRISTOPHER CROCE CATALIN STANCESCU
CONTENT EDITOR	LIN GOETZ
CUSTOMER SERVICE MANAGER	ANTHONY D. SPITZER
CUSTOMER SERVICE REPRESENTATIVE	MARGIE DOWNS
JDJ STORE 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

What would happen if the Internet went out of business?

BY JACK MARTIN

Once upon a time, one bought a computer and the software and then proceeded to use these for the purposes for which they were purchased – and this applied whether one’s system was small or large.

We now take the Internet for granted, much as we take water, electricity, gas, and the telephone for granted. Indeed, IBM has recently moved toward selling IT as a utility, with payment based on usage. However, the concept of the network as a utility took a large knock this month as Europe’s largest data network, KPNQwest, went into liquidation and WorldCom appears to be on the verge of going under as well.

While efforts are being made to rescue these companies, organizations using their networks, including many Internet service providers, are looking for alternatives. Forty percent of Europe’s Internet traffic passes through the KPNQwest fiber-optic network, so whatever happens may well have an impact on Internet traffic throughout the world.

WorldCom owns and manages the largest global network of its kind; it has complete control over the products and services it brings to market. This extensive global network coverage consists of more than 95,000 route miles and local coverage in 65 countries. Today, WorldCom’s UUNET IP network spans more than 3,800 Points of Presence (PoPs) throughout North America, Europe, and the Pacific Rim, and incorporates more than 2 million modem ports.

If the WorldCom and KPNQwest networks disappear, even for a short time, the effect will be very similar to the effects predicted for the Y2K “bug.” Companies may not be able to function.

WorldCom’s former chairman, Bernie Ebbers, spoke publicly about the company’s accounting problems at his church in Brookhaven, MS, his longtime home. At the end of the Sunday service he told the congregation, “I want you to know you aren’t going to church with a crook.”

Ask yourself, why did Bernie Ebbers make his announcement at a church instead of a major financial news network?

One could look at Ebbers’ background to get the answer. At one time, Ebbers ran one of the most successful companies in the telecommunications business – on paper. Who would have guessed that the native Canadian (who dropped out of college twice before graduating from Mississippi College) began work as a milkman by day and a bouncer by night? He started the long-distance resale company, LDDS (Long Distance Discount Service), and then started devouring smaller firms, including MFS Communications and UUNET. What would emerge was WorldCom, now one of the largest Internet carriers in the world and one of the biggest players in the telecommunications industry, boasting a set of financial records that are more fantasy than reality. Ebbers had no background working in telecommunications or computer science before LDDS.

He never slowed down on buying companies to take the time to learn to run what he had built. WorldCom grew by buying some 40 smaller companies.

In small towns, the milkman is always forgiven for his indiscretions, as everyone still needs milk. The major financial news networks are not as forgiving.

The NASDAQ is now back to where it was before the dot-com boom – just like the massive run up never happened. But something did happen: currently, every major corporation in the world and tens of millions of small businesses and consumers use the Internet on a daily basis.

The rise of the Internet tycoon bears comparison to the rise of the industrial tycoons of the 19th century. Consider this: Andrew Carnegie first invested in steel in 1861. Forty years later, after buying out a string of competitors, he sold his steel holdings to J.P. Morgan. This formed the core of the new U.S. Steel, making Carnegie the richest man in America.

There is, however, one great difference between the Carnegies and Rockefellers of the 19th century and the Internet tycoons of today. Carnegie effectively was accountable to no one but himself; today’s tycoons are servants of their shareholders. They will stay on top of the world only as long as their share price keeps rising.

So, it’s clear that we have a rocky road ahead of us, except perhaps for the 30% of small businesses in America that still don’t own a computer.

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

Things to know when installing WebSphere Portal

Into the Blue

BY JOE FARSETTA

In the past, I've written about operational readiness and operational excellence as performance models for data center environments. I'm an ops guy; a gearhead; the imperious overlord of all things within my domain. I'm the person you typically hand your space, equipment, applications, and business models to, and say, "Make it all work, then manage it."

ABOUT THE AUTHOR
Joe Farsetta is an engineer with more than 20 years of industry experience in telecommunications, networking, operations, business process architecture, applications, and support. An entrepreneur and inventor, Joe's past engagements have included Unilever, NJ Transit, and several key positions at Exodus Communications. He currently provides independent consulting services and seminars on in data center design, high-availability infrastructures, operational readiness, and service-level agreements.

E-MAIL
jjf10965@yahoo.com

The investment in the human aspect of a project is typically what will make or break any major IT undertaking. This asset can never be overlooked or taken for granted. It's where the rubber hits the road – where the planning, organization, and execution take place. POE, not Edgar Allen Poe, but Plan-Organize-Execute, is a surefire formula for success on any project.

This month I'll shift to the next level. Assuming all previous topics have been considered and holes have been plugged, it's time to take a giant step forward and build the application infrastructure. Your firm has decided to purchase and install WebSphere Portal. Where do you start? Let's begin with portals.

What's in a Name?

The word "portal" is overused – everyone has a portal, everyone needs a portal…it's like a fashion statement. What is a portal, anyway? What is its basic premise? Loosely translated, IBM defines a portal as an entryway that provides an end user with a secure, single point of interaction with diverse information, business processes, and other people, all personalized to the end user's needs. Not too shabby, but IBM gets even more granular than this. IBM believes a portal should provide:

- A single point of access to all resources associated with the portal domain
- Personalized interaction with the portal services
- Federated access to hundreds of data types and repositories, aggregated and categorized
- Collaboration technologies that bring people together
- Integration with applications and workflow systems

The Next Step in Portal Definitions: Horizontal and Vertical

Beyond the basic portal concept, IBM and other industry analysts got the notion that further granularity was needed to properly categorize portals. From those thoughts came the ideas of horizontal and vertical portals.

Horizontal Portals

A horizontal portal represents the primary infrastructure upon which the portal itself is constructed. It contains several layers and subsystems, including:

- ***Presentation layer:*** Includes a Web user interface along with pervasive device support
- ***Personalization:*** The ability to serve dynamic response to the user based on personal profiles
- ***Collaborative toolsets:*** Allow things such as e-mail to be exchanged
- ***Portlets:*** Provide the ability to easily attach software modules and other services to the portal
- ***Applications and workflow:*** Provide the matrices that allow for the integration of legacy and new applications
- ***Search and navigation tools***
- ***Publication and subscription toolsets:*** Provide the ability to author new content and publish it to subscribers
- ***Administration and security services***
- ***Integration toolsets and methodologies***

Vertical Portals

Vertical portals, on the other hand, cover a specific domain or business function. They may include functional areas within a company or industry and are typically built on top of a horizontal portal infrastructure. Think of the many pieces that make up an organization, such as sales, operations, IT, customer support, etc. Imagine each of these subsets as the vertical portals. As such, vertical portals are typically defined by the data, people, and processes they represent or serve.

Now that we understand what portals are and we've taken a peek at emerging standards of basic portal infrastructure and classifications, we can tackle what IBM has cooked up in the way of WebSphere.

WebSphere Portals

WebSphere Portal for Multiplatforms (WebSphere Portal) provides enhanced access to Web-based information and applications and provides a single access point to these applications, including associated content and processes. It comes in three flavors:

- ***WebSphere Portal Enable:*** The basic offering, which enables clients to build scalable portals that simplify and speed a user's access to personalized information and applications.
- ***WebSphere Portal Extend:*** An enhanced offering that allows portal users to act on information and applications accessed through collaborative efforts with other portal users. It includes all the capabilities of the basic offering, as well as instant messaging, extended search capabilities, and some analytical capabilities.
- ***WebSphere Portal Experience:*** The most robust of the WebSphere Portal offerings, it provides the capability for developing, deploying, and maintaining enterprise portals. This solution includes all the capabilities of the basic and enhanced offerings and adds some neat features to the mix, including application sharing, content management, and enhanced security features and functionality.

At this point, everyone on your team should be excited about implementing the project. Ensuring that the proper version of WebSphere Portal has been ordered is the first step. Now it's time to examine how the installation is done.

Supported Platforms

WebSphere Portal is supported on a variety of platforms and operating systems, basically the same AIX, Linux, Intel, Solaris, and Windows platforms that support WebSphere. Platforms and OS environments include the following, but keep in mind that the list represents the minimum requirements that must exist on the portal machine prior to installing Portal Server:

- ***RS 6000 hardware:*** AIX v4.3.3 or 5.1 (minimum)
- ***Intel platforms (IBM Compatibles):*** Linux RedHat
- ***Intel platforms (IBM Compatibles):*** Windows NT Server with fixpack 6a (minimum)
- ***Intel platforms (IBM Compatibles):*** Windows 2000 with service pack 2 (minimum)
- ***Sun platforms:*** Solaris 7 or 8 (minimum)

There you have it. Everything you need to know about what WebSphere Portal needs to run on – almost. Now come the subtle intricacies, including the critical applications that must be installed prior to the WebSphere Portal installation. These items were taken directly from IBM's documentation package. Let's look "under the hood" and see what powers WebSphere Portal from the standpoint of recommendations, considerations, and choices:

- ***IBM HTTP Server:*** IBM HTTP Server 1.3.19.1 is recommended. Portal Server supports all Web servers that the aforementioned level of WebSphere Application Server supports. If you plan to use a Web server other than Microsoft Internet Information Services (IIS), make sure IIS features are disabled before installation.
- ***WebSphere Personalization v4.01:*** The Setup Manager installs WebSphere Personalization in the same virtual application server where Portal Server is installed. To use an existing copy of Personalization, you must install it in the same virtual application server where Portal Server will be installed, prior to installing Portal Server. By default, Setup Manager installs Portal Server in a virtual application server named WebSphere Portal.
- ***Database and LDAP considerations:*** Depending on your configuration, your portal might require a relational database and LDAP directory. However, this software doesn't have to be installed on the same machine as Portal Server.
- ***DB2 Universal Database, v7.2 with Fixpack 5:*** During installation you can select to install the DB2 Server or DB2 Administration Client. After installation, a prompt asks you to install or not install the DB2 OLAP Starter Kit. Because WebSphere Portal doesn't include the OLAP Starter Kit, be sure to select "Do not install the OLAP Starter Kit," then continue along your merry way.
- ***Oracle v8.1.7:*** Although not shipped with WebSphere Portal, it can be used as your relational database. However, you must install Oracle prior to installing Portal Server. Also, if you've set up your database with a different name for the System Identifier (SID) and the Global Database Name, use the SID during the portal installation.
- ***IBM SecureWay Directory:*** SecureWay Directory v3.2.2 is recommended, but, and this is very important, in order to use SecureWay Directory with Portal Server, you must install SecureWay Directory, add a suffix, and then edit and import an LDIF file before you install the Portal Server component.

Above all, be careful. There are several versions and upgrades that aren't supported, including versions of DB2, HTTP Server, SecureWay Directory, WebSphere Application Server, WebSphere Personalization, and some versions of WebSphere Portal Server. A variety of Web browsers are supported, but not Netscape Communicator 4.7X. Before proceeding, it's best to check with your IBM rep or certified installation partner for the entire list of supported and unsupported products and applications.

Now, on to the physical stuff – your actual environment and what will influence it.

Physical Planning

The number of machines WebSphere Portal will require, along with the physical makeup and locations of these machines, depends on client requirements. Therefore, in-depth needs analysis and preengineering exercises need to be completed prior to installation. Portal installation and deployment are interdependent upon how WebSphere Application Server is built. As you can imagine, careful planning and a thorough understanding of what you're doing and where you're going is paramount.

Many resources and settings defined within WAS, such as Global Security settings, are shared across all applications, including the portal itself.

Make It a Double...

So what's it going to be? Will you install Portal on one machine or many? This is an important question to consider, since a single-machine install will need to include WebSphere Portal, WAS, and the Web server itself. This translates into a lot running on one box. Unless you're performing testing and development duties, this is probably the riskiest configuration for a production environment. Definitely not a design with any fault tolerance in mind.

Multimachine installations are far more robust and reliable for true production environments. Several example topologies for multimachine infrastructures follow:

- ***Multitiered topologies:*** Components such as the Web server, databases, etc., are installed across different machines.
- ***Vertical scaling topologies:*** Multiple portals, or portal processes, reside on a single machine.
- ***Horizontal scaling topologies:*** Multiple portals are created and deployed across more than one machine and rely on products such as Network Dispatcher, an HTTP redirector, to provide the modality of operation.
- ***HTTP server separation topologies:*** The HTTP server is located on a different machine than WAS and Portal.
- ***Demilitarized zone (DMZ) topologies:*** Commonplace in most Web-facing production sites, this topology utilizes firewalls to provide logical separation between machines within the infrastructure and the Internet. DMZs provide improved portal security and are the de facto standard where back-end processes and database resources are exposed.

Useful Tools

Some really useful tools are included in the product suite, one of which is the Setup Manager. Portal Server provides multiple software components that you can install. Each component has a variety of separate and distinct requirements; use the Setup Manager to install these components in a variety of configurations.

Depending on the installation you choose, the Setup Manager will prompt you for information during the installation. According to IBM's documentation, Setup Manager's installation options are broken into three offerings:

- ***Quick installation option:*** Uses configuration information stored in a response file to automatically install the Portal Server components. The response file is included in the software bundle. During installation, the installer enters the response file when prompted to do so by the Setup Manager. At that point, all components provided with the Portal Server will automatically install. No information input is required.
- ***Standard installation option:*** Uses configuration information stored in a response file to automatically install the Portal Server components. The response file is generated during installation and provides necessary information so you don't need to enter information during the installation. The response file preselects components that are installed on a single drive on a Windows environment or on default file systems on a Unix environment. The component interdependencies and subcomponent selections are managed for the user. With this installation some preconfiguration must be done. The installation defines what is in your response file and you can change these definitions. However, there is no validation for these changes during installation.
- ***Advanced installation option:*** You select the components you want to install. This is recommended for advanced administrators only because you provide the installation, configuration, and customization information using each component's installation user interface. Selected components can be installed on different drives; you're guided to make choices that satisfy component interdependencies, and you make the subcomponent selections. Because you're prompted for a lot of information during the installation, it's recommended that you gather your answers before beginning the installation process.
- ***Other Portal Server installation prompts:*** During installation of the Portal Server you'll be prompted for a Standard or Developer install option. Select the Standard installation if you want to install the security features included with Portal Server. The Developer installation doesn't use Application Server or a third-party authentication proxy to verify proof of identity.

I'm just scratching the surface here. Proper installation of Web Sphere Portal for Multiplatforms is complex. It requires genuine planning and attention to detail. Once the product is in use, prepare for the needs that come along with sustaining operations. Keep clean and complete documentation. It may save you some day.

Conclusion

Installation of WebSphere Portal is not for the faint of heart or novice. It requires real planning and understanding of the product, modalities, applications, resources, and project management skills. Intense up-front engineering, along with robust documentation and the use of planning and installation worksheets, should also be anticipated.

Remember the three key letters mentioned early in this article, POE. Adopt this as the engine behind your project plans. Use these three simple key elements in your WebSphere Portal design and implementation. Develop and manage the critical path. Surround yourself with individuals possessing knowledge, drive, and dedication. That's the investment in the human aspects of any major project. Believe me, it'll be worth the effort. In the end, those envious of your accomplishments and the success of your project may indeed nickname you "Edgar Allen."

An overview

Content-Centric Perspectives on IBM WebSphere Portal Family

BY MICHAEL T. FERRO & DOUGLASS J. WILSON

Portal has come to mean different things in different situations. Many enterprises approach a portal as a Web site enabling a partitioning of the displayable real estate into separate areas, each area having its associated content personalized to the needs and desires of the site visitor.

The content for the discrete areas is dynamically assembled into pages at runtime (see Figure 1). Such a portal could allow for delegated authorship rights of the content in its constituent areas, yet retain centralized management of access rights for portal visitors, centralized control of the overall site, and a host of other administrative features that make a site a portal. This is a very content-centric view of a portal and can be contrasted with a view in which the component sections are conceived of as applications (involving complex algorithmic interaction between user and site) rather than content (involving a more perceptual model of interaction).

The distinction between content-centric and application-centric portals is fuzzy, and in reality most real portals incorporate elements of both. Still, it's useful to consider the distinction because it explains a path taken by many enterprises in their initial application of portal technology. In this content-centric perspective, the problem exists in site layout and distributed control of site assets. It involves the ability to retain central control of sites and the content style, as well as some controlled or fixed part of the content. The central group needs to delegate the development and life cycle management of sections of content to other groups within the enterprise. This necessarily links the portal with a Web Content Management (WCM) system in some unique ways.

IBM WebSphere Portal Family Version 4.1 fits very well with the content-centric model. Its basic unit of extensibility, the portlet, is the natural mapping to the basic area described above. In WebSphere Portal Enable (the base version of the family), portlets are dynamically assembled into portal pages, which can then be further grouped into portal page groups (the approximate equivalent of sites). In WebSphere Portal Extend (the intermediate version of the family), the concept of places represents the grouping of portlets. Places represent an enhanced notion of pages and page groups, incorporating collaborative community elements in addition to increased layout flexibility.

IBM's strategy for addressing this content-centric approach to portals involves three parts:

1. An open, proactive approach with WCM partners to enhance the solutions available to customers
2. Inclusion of best-of-breed features and open interfaces for content handling
3. Active participation in the open-standards communities for defining and refining standards relating to content

Several new features of WebSphere Portal Family Version 4.1 involve content handling, including a new, flexible layout manager; a content organizer; a Web content publisher; and a new search capability. These features will be explained briefly, along with an exploration of the functional integrations between partner WCM systems and WebSphere Portal.

Content Development and Content Delivery

Before exploring these components, it's important to consider the systems context in which enterprise portals operate. In addition to a strong environment for development and contribution of content to a site, enterprises require robust delivery systems for sites, including serious infrastructure for security, capacity and load balancing, metrics, complex network topologies, and other elements. Figure 2 provides a simplified view of this.

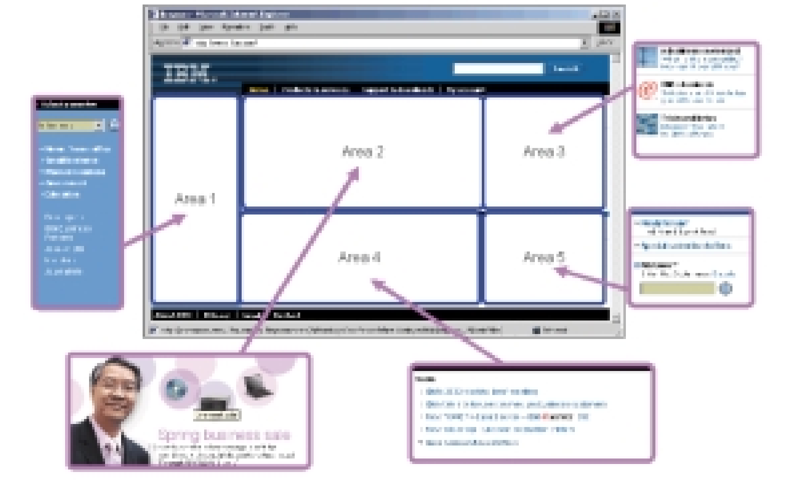


FIG. 1: WEB PAGE WITH FIVE AREAS

The vertical line in the figure illustrates the separation of content-contribution infrastructure from contentdelivery infrastructure. WCM systems are traditionally strong in the content contributor part of the picture. Web application servers and related server technology, including WebSphere Portal Family, are strong in content delivery. Because portal servers involve the dynamic assembly of content elements, several issues arise that blur this boundary. For example, site layout, traditionally part of the WCM sphere, is now part of the portal functionality because the layout must include contextual elements relating to security (for example, rights prescribing who can access, edit, or manage the individual layout elements or portlets).

One irony is that some so-called portal server products, by concentrating on the content-contributor sphere without full regard for the content-delivery infrastructure, can appear to present a paradigm of great flexibility for the development of content-centric sites. This is explored more fully in this article as "site builder WCM systems." Such systems are largely extensions of WCM systems rather than true portal builders because they leave the centralized management of access and user community out of their solutions. More important, they don't address the need for robust delivery systems with the inherent complexities of an enterprise network. In contrast, WebSphere Portal operates in a closely integrated fashion with WebSphere Application Server, allowing participation in the full range of configuration choices in the context of enterprise networks. At the same time, through partnerships, WebSphere Portal embraces integration with the emerging set of portal site builders, offering a robust solution that addresses both aspects of enterprise requirements.

Also note that in real systems there are typically multiple content repositories (rather than the single repository shown in Figure 1). In particular, content other than Web content, such as digital images or other enterprise content, is typically accessed via a portal. Such capabilities are supported by IBM Content Manager and its applications, including Content Manager OnDemand, Content Manager ImagePlus, and Content Manager VideoCharger.

Integration Patterns for WebSphere Portal plus WCM Systems

When WebSphere Portal is integrated with WCM systems (including partner WCM systems and IBM's Web Content Publisher), several distinct integration patterns are evident.

PUBLISH CONTENT TO PORTAL

This involves using the WCM system to develop content and keep a master copy of content in the WCM domain. An explicit step is then employed ("publish" or "deploy") to make the content available to the live site and its users. When the live site is a WebSphere Portal site, this essentially means that portlets can access the content and render (or "play") it. IBM has produced a set of integration kits for a set of partner WCM systems demonstrating this capability for a particular content model. The kits include a "how-to" document resembling a redpaper. The content model is one of Corporate News for a fictitious company and utilizes open XML standards (OCS, RSS, and NITF). The kits can be found by entering "Web Content Management" in the Portlet Search field at www7b.software.ibm.com/webapp/portlets/portletemarketplace. Kits illustrating integration with Interwoven, Vignette, and Documentum are available, and others are under construction.

LAUNCH WCM FROM PORTAL

This integration pattern allows content contributors to access the WCM system from inside the context of the portal. It involves either incorporating the WCM system's user interface as a portal application (portlet), or at least launching the WCM system's function from a portlet. It typically involves sharing the user identity to achieve a single sign-on effect, such that a portal user (who's already identified himself and been authenticated) need not sign on to the WCM system separately – his or her WCM identity was already established at portal login time. This integration pattern between a WCM system and WebSphere Portal is especially important for WCM systems that incorporate significant runtime functionality (e.g., dynamic content delivery systems). Most of IBM's WCM partners have produced this style of integration package with WebSphere Portal Family.

WCM MANAGING PORTAL CODE CONTENT

This type of integration introduces additional complexity and is charac-

ABOUT THE AUTHOR
Michael T. Ferro is a senior software engineer at IBM Corporation working in the WebSphere Portal development organization. Mike is the lead architect for the integration of Web Content Management with WebSphere Portal.

E-MAIL
ferromt@us.ibm.com

ABOUT THE AUTHOR
Doug Wilson is a distinguished engineer working on the Architecture and Strategy Team in the WebSphere development organization. Doug's major role is to provide strategy and engineering guidance to components of the product line for delivery of content and applications.

E-MAIL
douglas_wilson@us.ibm.com

terized by a consideration that code elements (servlets, JSPs, etc.) can be viewed as just another form of content asset for a site. When compared to other content assets, development of code assets requires a distinct skill set, as well as considerably different processes for deployment. Portal servers, including WebSphere, introduce specific administrative requirements (beyond what's required to deploy applications on an application server alone) for deploying code and applications. The additional functionality provided by the portal server requires that additional configuration information be provided.

As these deployment techniques mature and standardize (e.g., JSR 168 and WSRP), tooling support will likely expand. In addition to approaching this space from a WCM perspective, most enterprises find it useful to manage code content in a Source Code Management (SCM) system. If utilizing a WCM system for common versioning and control of all site assets, they then export from the SCM system and import into the WCM system. IBM's tooling is based on Eclipse standards, utilizing CVS repositories. Some WCM partners are approaching alternative functionality in this pattern.

INCORPORATING ADDITIONAL RUN-TIME COMPONENTRY

One example of this is personalization technology, which provides the portal user with an experience that matches content to users. IBM offers two types of personalization technology: a rules-based engine and a recommendation engine. Both are included in IBM WebSphere Personalization, which is packaged with WebSphere Portal Family. The technology works by matching attributes of the portal user with specific content (examples are provided with the WebSphere Portal Family). The content collection must contain appropriate attributes, or metadata, to enable the matching technology. The user object contained in WebSphere Portal, WebSphere Member Services, is already enabled for the WebSphere Personalization technology (by implementing the user resource Java interfaces), out of the box. Additionally, some partner WCM systems provide native support to enable the deployment of content collections enabled for personalization.

Another example of this pattern is integration with servers at the edge of the network, often applied as part of a caching strategy for high-usage sites. As content elements within the scope of an individual portlet are modified, the portlet then becomes the control point for invalidation of the edge cache. A new prerendering event action in the revised Portlet API enables appropriate cache invalidation logic to be inserted and controlled by the portlet.

WCM systems that provide dynamic content delivery, which is a runtime function, also fit this integration pattern. In cases where such WCM systems have customers with existing content delivery applications (CDAs), it's possible to intercept those content streams through generic access portlets. Such portlets would need to be written to recognize the specific content models of the existing CDAs. This would enable existing content collections managed by such WCM systems to be displayed within the portal, along with the attendant advantages of access rights administration and other runtime portal advantages.

INCORPORATING WEBSPHERE PORTAL ABSTRACTIONS INTO WCM SYSTEMS

As mentioned earlier, there is a significant set of new abstractions that are formalized by portals. When these are incorporated into a WCM system's set of asset types, the combined capability can result in great power. This type of integration pattern is beginning to emerge among IBM's WCM partners.

Site Builders as Extensions of WCM

Some WCM systems extend the abstraction of a site in such a way that they resemble site builders for content-driven sites. In these systems, adding the ability to define and deploy content-driven portlets, which can be administered as can any other portal object, is very powerful. This is feasible with WebSphere Portal because of its separation of an abstract portlet (which includes the code to implement the portlet) from one or more corresponding instances of concrete portlets (which couple the code in the abstract portlet with a full set of configuration parameters, resulting

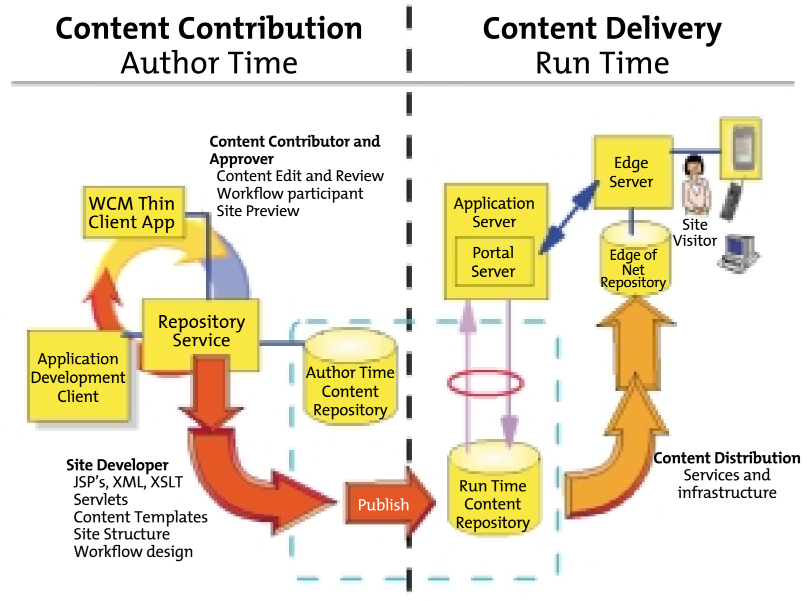


FIG. 2: SIMPLIFIED CONTENT-CENTRIC ARCHITECTURE

in a customized executable instance of a portlet). Thus, using the flexible content asset–definition capabilities present in most WCM systems, content plus portlet configuration parameters can be defined as an asset–type and deployed as concrete portlet instances. All that's required is an appropriate abstract portlet to recognize the applicable content model. IBM has written several such generic portlets as part of its collection of building-block portlets. An expanded set of these content pattern portlets is of considerable interest to IBM, and customers and will likely see expanded capabilities in the future.

This idea can be extended beyond portlets as well. Site builders can incorporate abstractions for PortalPage, PortalPageGroup, PortalSkin, and PortalTheme. The first two are useful in the development of "places" in a portal (a collection of portlets, PortalPages, or PortalPageGroups that fit together as a logical section of a site). PortalSkin is useful for cosmetic look-and-feel elements in the framing of portlets, and PortalTheme is useful for user-interface issues such as layout format and the style of navigational elements. Automated deployment of these portal-specific content assets requires sophisticated support in the site builder. Fortunately, most WCM systems incorporate the capability for customers to flexibly define their own asset types. By incorporating such portal abstractions into site builders, business users are enabled to directly manage entire portal sites, or extensive sections of portal sites, while still retaining the benefits provided by the portal server infrastructure (single sign-on, secure access to information and applications, centralized administration of users, etc.).

While complete systems to realize the full potential of such portal site builders are in the future, significant parts of this can be accomplished today by coupling a WCM system with WebSphere Portal. IBM has signaled its intent to continue in this direction.

Information Access Rights

Another abstraction formalized by WebSphere Portal is the management of access rights to applications and to information. Clearly, there's some opportunity to extend the concept of access rights already inherent in the content management domain, to map more explicitly and naturally to the portal server's runtime management of these rights. Such integrations between WCM systems and WebSphere Portal are largely future considerations.

New Content-Handling Features in WebSphere Portal Family 4.1

Another set of integration patterns can be described related to some of the new content-handling features in WebSphere Portal Family Version 4.1.

NEW LAYOUT MANAGER

The new layout manager is a significant upgrade from previous versions of WebSphere Portal. Individual portlets can now be placed in arbitrary spots on a page and sized by specifying either explicit sizes or percentages. Portal themes, which include the appearance of navigators, can be applied at the page-group level or more broadly. Skins (the stylistic "framing" elements) can be applied at the level of individual portlets, ranging from no skin to a number of choices. IBM provides an expanded set of skins and themes, and an easy interface allows customization of skins. The net effect is that even in the base level (WebSphere Portal Enable), the layout capability is flexible enough to allow virtually any layout appearance. In the intermediate level (WebSphere Portal Extend) and higher, additional layout flexibility is provided with places in the form of layout templates. As mentioned earlier, places also provide collaborative community-oriented elements.

CONTENT ORGANIZER FEATURE

Many portals introduce the notion of content that's directly managed by the portal, or in some sense internal to the portal. The content-organizer feature represents IBM's approach to this idea. It provides the ability to view a content collection in a hierarchical folder–like view and define communities with access rights on the given collection. For example, a content collection relating to a particular topic (such as information and documents relating to a specific set of chemical analysis) can be shared among a community of portal users with an interest in that particular topic (such as the set of chemists in a given enterprise who are sharing and discussing the collection).

IBM's approach to such content viewing and sharing was to implement this as a federated interface for content collections rather than as a specific internal repository. A given content collection is registered to the portal by implementing a set of Java interfaces for the content and then going through an administrative registration step. This is the same set of interfaces used by WebSphere Personalization (the content resource interfaces), which represent WebSphere's abstraction for content. Content that's registered with the content organizer is also enabled for personalization. Together with this federated interface for content collections, IBM provides a pair of generic viewer portlets that understand how to traverse the collections.

SEARCH FEATURE

WebSphere Portal Family Version 4.1 includes a powerful new built-in search capability. This feature is optimized for situations involving medium-sized content collections requiring extremely accurate result sets. Content registered with the content organizer is automatically enabled for the crawler, and there's an interface for indexing external collections as well.

IBM's search-broker technology (Lotus Extended Search) continues to be included with WebSphere Portal Family Version 4.1 as well, as it was with previous versions.

Web Content Publishing

This feature provides WCM functionality and is included with the portal. It integrates with WebSphere Portal Family in the ways described above, much as IBM's partner WCM systems integrate. (See article on this in ***WSDJ,*** Volume 1, issue 5.)

What Lies Ahead

IBM WebSphere Portal Family provides for dynamic assembly of content elements, with centrally controlled portal layout management coupled with the access rights management. This represents a powerful base to start with. IBM is further expanding its activities in this area. Expanding the interfaces to allow external management of the portal model (the structure that defines logical assembly of portlets into pages and page groups) would enable WCM systems to build and manage complete portal sites. Expanding the versioning capabilities of the portal model, internally and externally, would similarly enhance customers' ability to manage portals from WCM systems. Providing an expanded set of content patterns, and the portlets to make them realizable, would further enable the assembly of custom portal sites by content professionals rather than programmers. As the standards relating to content solidify, look for IBM to be ready with support for those emerging standards.

Summary

A content-centric portal is one where site visitors interact with the portal via a perceptual model. Site layout and delegated control of site assets typify the requirements for establishing and managing such portal sites. WebSphere Portal Family Version 4.1 provides a considerable technology base of available solutions for portal customers taking this content-centric approach. Integrations with partner WCM systems are a key part of this, as is active participation in standards activities relating to content and inclusion of strong content-handling features. IBM is aggressively continuing in this direction and expanding the portfolio of solutions available to customers taking a content-centric view of portals.

"A content-centric portal is one where site visitors interact with the portal via a perceptual model"

An easy-to-implement scheme

A Programmatic Approach to On-the-Glass Integration of Portlets

BY SAMAR CHOUDHARY, SHANKAR RAMASWAMY, SAI RATHNAM, AND AMBER ROY-CHOWDHURY

This article describes an easy-to-implement scheme for user-driven, on-the-glass integration of portlets using features of the WebSphere Portal Server (WPS) portlet programming model. Users can transfer information displayed on one portlet to another portlet on the same page, which then acts on it and refreshes the display with new content. Portlet developers can select specific fields in their portlets as useful to other portlets within the same page. For example, a travel-booking portlet exposes the name of the destination city as a source of information. A weather portlet within the same portal application can then be asked to display weather information for that city. The entire interaction is driven visually on the browser window with a user interface model.

ABOUT THE AUTHOR

Samar Choudhary is an advisory software engineer at the IBM Pittsburgh Lab. Samar has developed portions of the WebSphere Application Server and is currently involved in WebSphere Portal Server development.

E-MAIL

samar@us.ibm.com

The scenario we'll use has been modified to support such user-driven integration of diverse portlets. We'll discuss the design of the application as well as the portal features and additional technologies that are exploited to achieve the integration. The portlet APIs used in the examples pertain to WPS version 4.1.

Scenario: Order and Shipment Tracking

Acme Corporation is an online retailer that receives requests from customers for information about orders. Acme wants to empower their customer service representatives with an "Order Status" portal page that will provide a single point of access for all order-related information. Using this page, Acme's customer service representatives will be able to quickly provide customers with accurate order-related information.

Acme builds an Order Status portal application containing a number of different portlets. The portlets connect to one or more back-end systems to provide information about customer orders as described below:

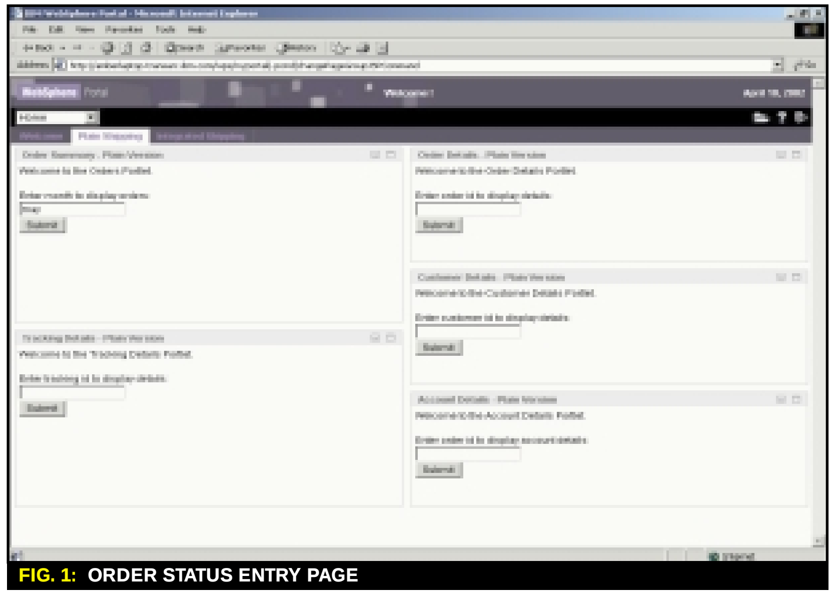
- ***Order Summary:*** Connects to Acme's order-tracking back end to provide a listing of all the orders for a particular month. Each entry in the list contains an order ID and a customer ID.
- ***Order Details:*** Connects to Acme's order-tracking back end to provide order details corresponding to a specific order ID.
- ***Customer Details:*** Connects to Acme's customer relationship back end to provide customer details corresponding to a specific customer ID.
- ***Tracking Details:*** Connects to an external vendor site to provide tracking details corresponding to a tracking ID.
- ***Accounts Receivable:*** Connects to Acme's financial back end to provide information on the accounts receivable information corresponding to a specific order ID.

A view of the "Order Status" entry page is shown in Figure 1. The user interacts individually with each portlet; it's possible that the portlets were developed independently. For example, the Tracking Details portlet might access the database of an external shipping company and might have been provided by that company, while the remaining portlets might access the databases of various departments within Acme and might have been developed by Acme.

To retrieve complete details about a particular order, an Acme customer service representative enters various identifiers displayed in one portlet view into other portlets' entry fields. After a series of interactions, each portlet retrieves and displays details about some aspect of a particular order. Figure 2 shows the Order Status portal page after a series of interactions has been completed to display details about various aspects of a particular order.

Acme wants to facilitate interaction among the Order Status portlets by enabling easy transfer of compatible data between them. For example, Acme wants to enable a customer service representative to click on a specific order ID and, through a menu, have the option of passing that order ID to the OrderDetails portlet, which will react by displaying specific details about that order. Such preconfigured interaction options between portlets would eliminate the manual observation and entry of bits of data from one portlet's view into another's entry field, and would also simplify the end user's learning process, eliminating consequent errors.

A menu-driven interaction for determining compatible interactions is shown in Figure 3. When the user clicks on the arrow icon next to an order ID in the Order Summary portlet, as shown in the figure, a list of target portlets that could react to the order ID are displayed in a menu. In our example, the Order Details and Accounts Receivable portlets are valid destinations for an order ID. When the user selects a particular target, the data contained in the source portlet (in this case the order ID) is passed to the target portlet. The target portlet then queries its back-end system using the given order ID and refreshes its display with information relevant to that order ID.



Order Summary - Plain Version

Welcome to the Orders Portlet.

Enter month to display orders:

May

Submit

Order Details - Plain Version

Welcome to the Order Details Portlet.

Enter order id to display details:

Submit

Customer Details - Plain Version

Welcome to the Customer Details Portlet.

Enter customer id to display details:

Submit

Tracking Details - Plain Version

Welcome to the Tracking Details Portlet.

Enter tracking id to display details:

Submit

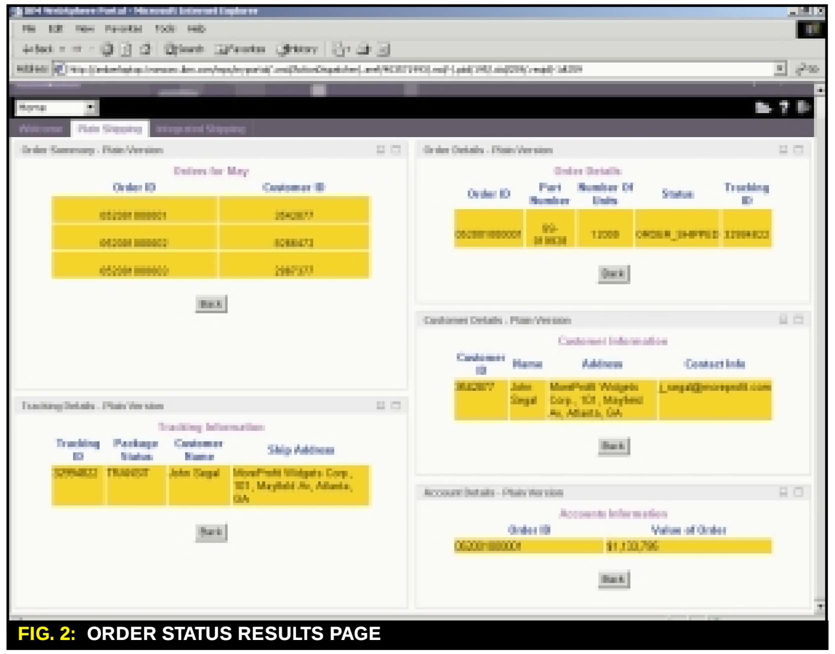
Account Details - Plain Version

Welcome to the Account Details Portlet.

Enter order id to display account details:

Submit

FIG. 1: ORDER STATUS ENTRY PAGE



Order Summary - Plain Version

Orders for May

Order ID	Customer ID
052001000001	2542877
052001000002	8288473
052001000003	2987377

Back

Order Details - Plain Version

Order Details

Order ID	Part Number	Number Of Units	Status	Tracking ID
052001000001	99-918838	12000	ORDER_SHIPPED	32994822

Back

Customer Details - Plain Version

Customer Information

Customer ID	Name	Address	Contact Info
3542877	John Segal	MoreProfit Widgets Corp., 101, Mayfield Av, Atlanta, GA	j_segal@moreprofit.com

Back

Tracking Details - Plain Version

Tracking Information

Tracking ID	Package Status	Customer Name	Ship Address
32994822	TRANSIT	John Segal	MoreProfit Widgets Corp., 101, Mayfield Av, Atlanta, GA

Back

Account Details - Plain Version

Accounts Information

Order ID	Value of Order
052001000001	$1,133,796

Back

FIG. 2: ORDER STATUS RESULTS PAGE

Design and Implementation

BASIC APPLICATION DESIGN

The design of the sample application follows the traditional MVC approach for developing portlet-based Web applications. Each component portlet is associated with a portlet class implementing the ActionListener interface from the portlet API. The control logic is encapsulated in the portlet class. Each component portlet is associated with a JavaBean for accessing and encapsulating data from a back-end system. The JavaBean represents the model and is passed by the portlet to a JSP, which renders a view by accessing data from the bean. Listing 1 (code for this article can be found at www.sys-con.com/websphere/sourcec.cfm) shows relevant portions of the Order Summary portlet code. Listing 2 shows relevant portions of the JSP used to render the Order Summary view.

Changes to Enable User-Driven Interaction

Enabling on-the-glass interactions involves modification to the view-generation code in the JSPs and the action-processing code. In the view-generation code in the JSPs, additional markup and JavaScript utilities are needed to facilitate the selection of sources and targets by the user. During action processing, additional functionality is needed to ensure that the selected target receives and processes the action selected by the user. Figure 4 illustrates the steps involved in facilitating one of these interactions – sending an Order ID from the Order Summary portlet to the Order Details portlet. This is a concrete example; other such interactions may follow a similar design pattern.

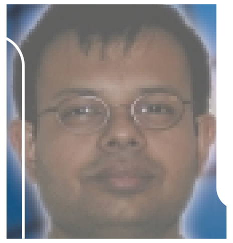
1. Within the doView() implementation of the Order Summary portlet, portlet action URIs corresponding to each order ID are created. Listing 3 illustrates the process of creating the portlet action URIs. Note that two action URIs are created for each order ID – one action sends a message containing the order ID to the Order Details portlet, and the other sends a

ABOUT THE AUTHOR

Shankar Ramaswamy is a senior software engineer at the IBM Pittsburgh Lab. Shankar is currently involved in designing and developing technologies for standards-based front-end integration.

E-MAIL

shankar4@us.ibm.com



ABOUT THE AUTHOR

Sai Rathnam is an advisory software engineer at the IBM Pittsburgh Lab. Sai has developed portions of the WebSphere Application Server and is currently involved in assisting WebSphere business partners with their design and development needs.

E-MAIL

rathnam@us.ibm.com

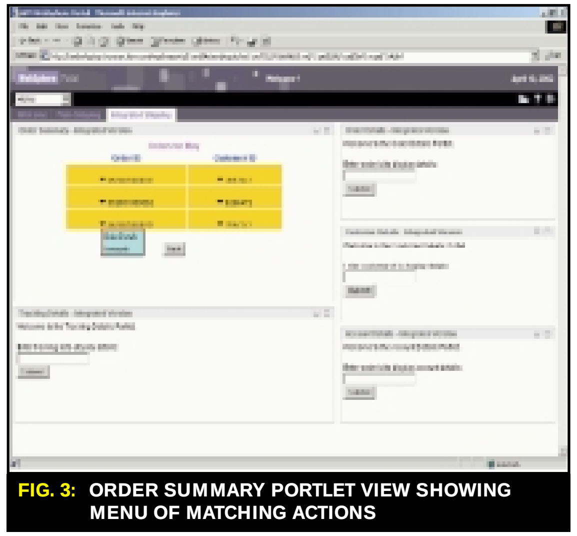


FIG. 3: ORDER SUMMARY PORTLET VIEW SHOWING MENU OF MATCHING ACTIONS

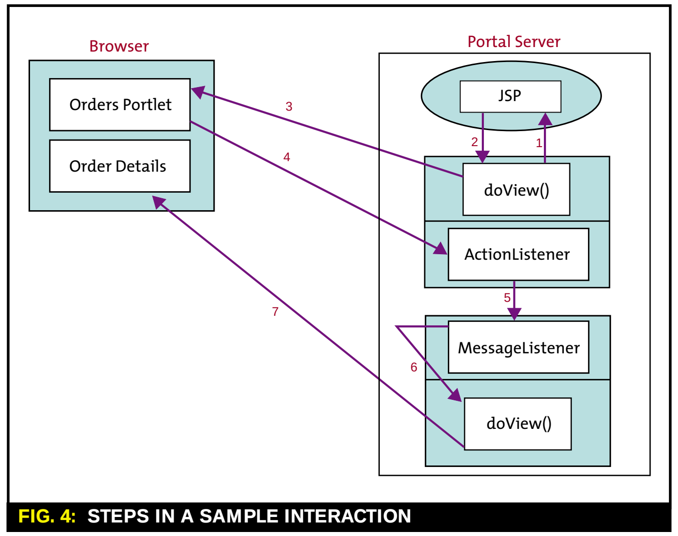


FIG. 4: STEPS IN A SAMPLE INTERACTION

message containing the order ID to the Accounts Receivable portlet. These are the two actions on the Order Summary page that can process an order ID. Other actions on other portlets that can be triggered by other data, such as the customer ID, can be treated in a similar manner.

2. The action URIs are set on the JavaBean, from where the JSP to render the view can retrieve them. The set of action URIs is then added to the JavaBean and passed to the JSP responsible for creating the view of the Order Summary portlet. The JSP retrieves the action URIs from the JavaBean. An image is inserted for each source (each order ID in this case), which we'll refer to as a "clickable icon." A JavaScript function is associated with the onClick event for the clickable icon. The function is used to display a menu and associate an action with each menu item. Each action receives the order ID and sends it to a target portlet that can process the order ID. Listing 4 shows the JSP for encoding the action URIs in the menu (the relevant portions of the code appear in bold).

The JavaScript functions for the menu are defined in a library, which must be included in the page header. This can be done by one of the application's portlets in its beginPage operation.

3. The Order Summary portlet now completes its doView(). The portlet container assembles the entire page and sends it back to the browser, which displays it. In particular, the Order ID and Customer ID columns are displayed as part of the Order Summary portlet's view. The clickable icon inserted by the JSP appears next to the order ID in each row.

4. If the user clicks on any of the clickable icons, the JavaScript function is executed to display a menu of matching actions. If the user selects an action from the menu, it results in the actionPerformed method being invoked on the Order Summary portlet. The name of the action specifies to the portlet that it needs to send a message containing the order ID to the target portlet. Listing 5 shows the action-handling code for the Order Summary portlet (the additions to handle the sending of the order ID to target portlets appear in bold). Note that there are two new actions, one for sending the order ID to the Accounts Receivable portlet, and the other for sending it to the Order Details portlet. Also, a custom MessageData object is used to encapsulate the message data and convert it to a string representation for sending on the message; the application can use any suitable message object for this purpose.

5. The message is received by the Order Details portlet, which extracts the message name and order ID from the message and sets them on the portlet session for retrieval by the subsequent doView() method. This is illustrated in Listing 6.

6. The doView() method on the Order Details portlet is invoked by the portlet container, which then retrieves the action and order ID information set in the portlet session by the message listener code and proceeds to retrieve and display the order details. Note that where the information was set in the session is immaterial at this point. In particular, it could also have been set in the action listener instead of the message listener if the user interacted directly with the Order Details portlet instead of indirectly through the Order Summary portlet. Listing 7 shows the relevant portions of the Order Details portlet's doView() method.

7. The portlet container finishes calling the doView() methods on each of the portlets on the page and assembles and sends the new page back to the browser. The browser's new page contains the details of the specified order ID in the Order Details portlet's view.

Conclusion

We've discussed a programmatic approach to enabling portlets for on-the-glass integration. The approach is based on identifying matching actions on target portlets, providing message-passing support for invoking such actions from source portlets, and using JavaScript to give end users visual cues for the matching actions in the form of a menu.

ABOUT THE AUTHOR
Amber Roy-Chowdhury is a senior software engineer at the IBM Pittsburgh Lab. Amber has developed portions of the WebSphere Application Server and is currently involved in WebSphere Portal Server development.

E-MAIL
amberr@us.ibm.com

Target content to your visitors

Personalization Within WebSphere Portal Server

BY ROB WILL

Web site personalization is a powerful tool that enables easier portal maintenance while helping you reach your goals. Personalization systems enable business users to target Web site content to the visitors who come to their sites. Targeting can be based on virtually any characteristics, including the visitor's role, classification, preferences, and the service level they're entitled to.

ABOUT THE AUTHOR

Rob Will has been part of WebSphere Application Server and Studio tools development since WebSphere's first release. Rob's current focus is providing tooling that helps WebSphere customers create and target content for their Web sites.

E-MAIL

willrc@us.ibm.com

Personalization is often confused with simple customization, which site visitors use to select the categories of information they want to see. With most personalization solutions, site owners use business rules or other techniques to present visitors with information to maximize the success of each visit. WebSphere Portal Personalization allows business users to match site content to visitors using either business rules or collaborative filtering.

The Personalization component contains three major runtime subcomponents (the rule engine, the resource engine, and the recommendation engine) and two major development-time components (the Personalization workspace and a set of WebSphere Studio wizards.) The rule engine and recommendation engine provide support for matching content to site visitors based on either explicit rules (rule engine) or collaborative filtering (recommendation engine). The resource engine provides an adapter layer that allows customers to define various user and content repositories to the Personalization system. The Personalization workspace is a browser-based tool that allows business users to define how content should be matched to the site. Programmers use WebSphere Studio wizards to define the adapter classes required by the resource engine and content spots used to render the content within portlets and JSPs.

How Personalization Works

There are four basic steps in personalization. The first is to identify the site visitor. With the Personalization component, users can be identified in a variety of ways. You can rely on WebSphere authentication or custom authentication to identify the visitor, or you can rely on a cookie. While many sites choose authentication, it's not necessary for personalization.

The second step is to retrieve the visitor's profile, which is the information the site stores about each visitor. This information can be stored practically anywhere. It might be in LDAP, a database, or one or more programs – the Personalization component allows you to store it in any of those places and more. The only requirements are that it be accessible with Java and that you can retrieve the information based on a unique key (the user ID or another unique token).

The third and most visible step is to select the content. This can be done in a variety of ways. The Personalization component provides two distinct matchmaking services: rules and recommendations. Rules are an explicit directive of how content should be selected for each user. For example, "If the user is interested in active vacations, then select white-water rafting specials."

Recommendations are also available; The personalization component makes recommendations based on a set of algorithms collectively known as collaborative filtering. Collaborative filtering matches the user's current and previous behavior with other users' previous behavior.

The last step is to present the content within the Web page. The Personalization component's presentation support leverages WebSphere's user interface frameworks, JSPs, and portlets to render the personalized content in the correct way using the correct markup language.

The Personalization Component

Typically, site owners start to think about personalizing their site after it's up and running. They look at a banner ad or a list of news articles and decide that their site would be more effective if the ad or list were optimized to be meaningful to the site visitor or, more important, to lead the site visitor to the desired end result.

To understand how personalization is implemented within WebSphere Portal, we'll look at the steps in reverse order, starting with the presentation of the content.

PRESENTING THE PERSONALIZED CONTENT

Having picked a part of the page or a portlet to personalize, the first step is to "make a hole" in the site by removing static content and replacing it with a placeholder for content that will be returned via personalization. A placeholder is needed because when the portlet or page with the ad is being developed, the actual content hasn't yet been determined. As shown in Figure 1, the page is viewed as a number of placeholders at author time, but is filled with the appropriate content at runtime.

Since presentation within the Personalization component is based on standard JSPs, the placeholder is a standard useBean tag and corresponding scriptlet for displaying the content returned by the bean.

After identifying where the personalized content should appear, you must decide what kind of content will go in each spot. For example, one placeholder might be for an ad image with a link to another page, another for a table listing three product recommendations, and a third for a list of news stories that will appeal to visitors. When you know the content type each spot will display, you can create the required JavaBean by running the Content Spot wizard within WebSphere Studio Application Developer (WSAD). The Content Spot wizard is very simple. Just fill in the name you want to assign to the new content spot and choose the content type that will be returned. For this discussion, we'll assume the content spot is called Confidential-NewsSpot and the content type returned is Company-News items.

After creating the content spot bean, simply drag and drop it onto the page where the content will be displayed (if you're using a JSP editor that doesn't support drag-and-drop of JavaBeans, you can manually create the useBean tag). The useBean tag defines the bean instance on the page and enables you to format the results from the bean to display an image, table, or something else by writing simple scriptlet logic or using WSAD PageDesigner's WYSIWYG controls. Since the content spot created by the Content Spot wizard returns typed resource objects, PageDesigner can help the page creator choose the properties to fill in each field within the page (see Figure 2).

We've assumed that a resource class representing the desired content type already exists. For simplicity, we'll continue with that assumption for now and will discuss later how the resource class was created. What's important to know is that each content type is represented by a resource class, and the properties of the resource class determine what values can be displayed within the page as well as what properties can be used for selecting the content within rules.

CREATING A RULE

At this point, the site is ready to call a rule to determine what content should be shown and to format and display the returned content,

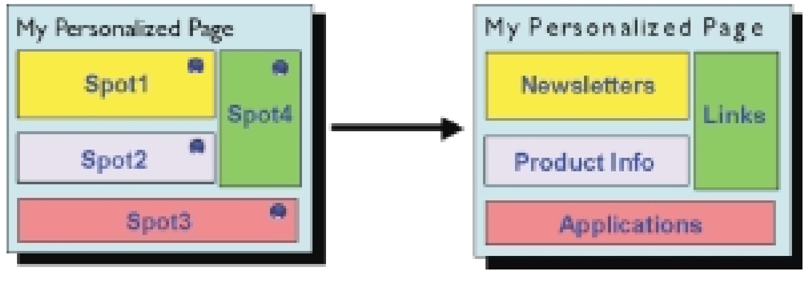
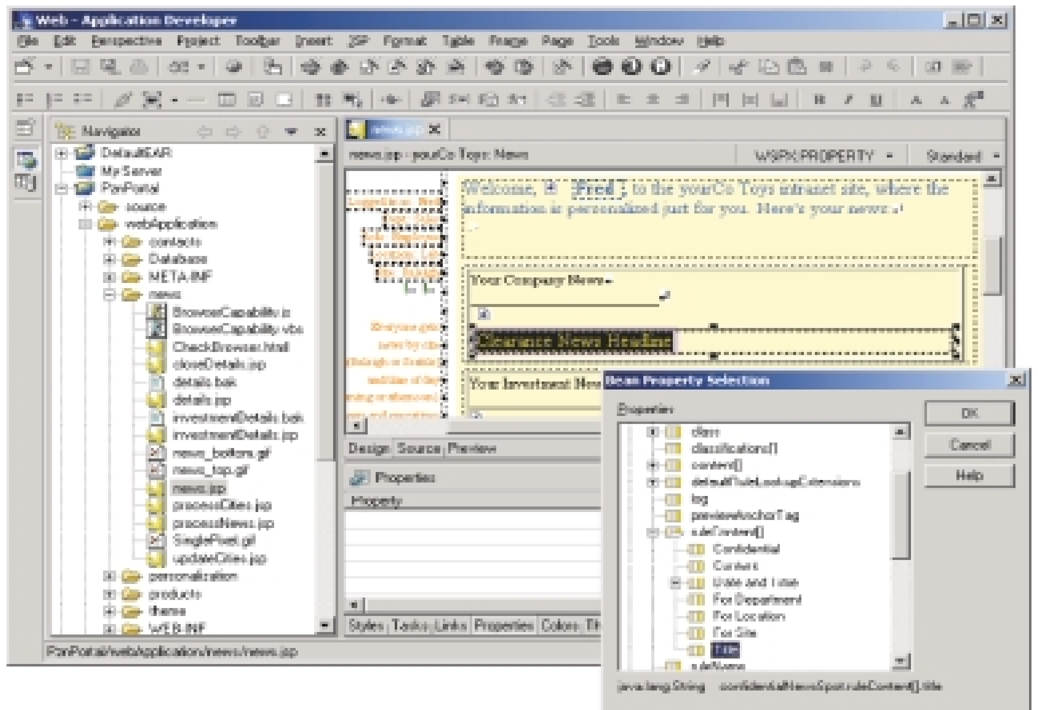


FIG. 1: WEB PAGE AT AUTHOR TIME AND RUNTIME



Welcome, Fred, to the yourCo Toys intranet site, where the information is personalized just for you. Here's your news:

Your Company News

Clearance News Headline

Your Investment News

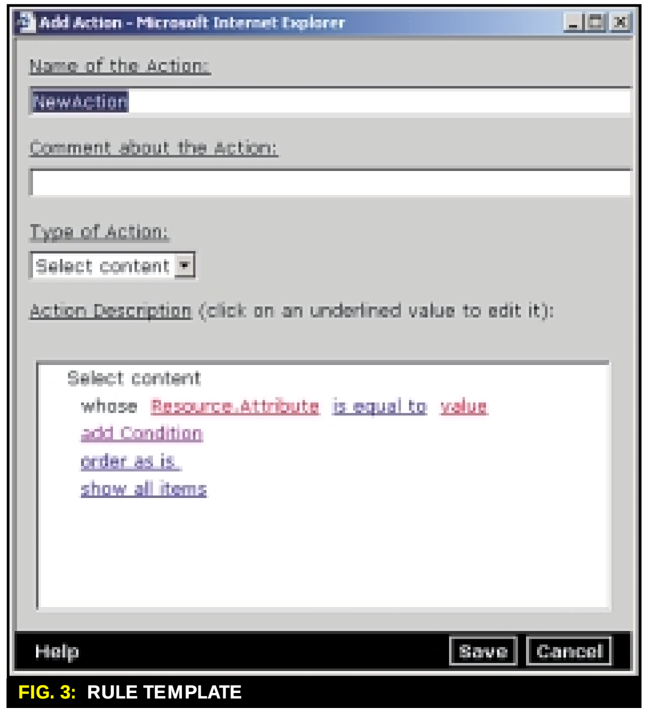
java.lang.String confidentialNewsSpot.ruleContent[].title

FIG. 2: PROPERTY SELECTION WITH PAGEDESIGNER

but the actual rule hasn't yet been defined. The rule is created using the Personalization workspace, a browser-based tool designed to allow business users to create campaigns and rules, as well as to preview the site under a variety of what-if scenarios.

Before creating a specific rule to fill in the CompanyNewsSpot, it's useful to consider the basic concepts implemented by the Personalization component. The Personalization component rules have three basic concepts:

1. ***Classification:*** Classifier rules classify the site visitors. For example, a classifier rule might specify users as being entitled to silver, gold, or platinum service.
2. ***Action:*** Action rules select or update content. For example, an action rule might select all the CompanyNews articles intended for the current visitor's location, or might update the user profile's list of articles read in order to avoid showing the same article more than once.
3. ***Binding rule:*** Binding rules combine classifiers and actions to perform conditional logic. For example, a binding rule might specify that users with confidential clearance be shown the confidential CompanyNews and the nonconfidential news, while users without confidential clearance are shown only the nonconfidential news.



Name of the Action:

NewAction

Comment about the Action:

Type of Action:

Select content

Action Description (click on an underlined value to edit it):

Select content
whose Resource.Attribute is equal to value
add Condition
order as is
show all items

FIG. 3: RULE TEMPLATE

Rules are created using a "point-and-select" metaphor. Each rule type has a specific rule template, as shown in Figure 3.

After the template is filled in with the appropriate selections, the corresponding content will be returned to the content spot within the JSP.

IDENTIFYING THE USER AND RETRIEVING USER INFORMATION

When a user signs on to the Portal Server, his or her authenticated ID is used to retrieve the user profile. The user profile defined within Portal can be used to access user information for use within rules because it's a resource class. If you have additional user information stored in a different repository (such as another database or one or more application repositories), you can define additional user profile classes to the resource engine, making the additional information available to your rules. If the primary key used to retrieve user information from the other user repository matches the user's authenticated ID, then you're finished. Otherwise, you need to create a mapping class to tell Personalization which ID to use when retrieving additional information.

Additional Features

CAMPAIGN MANAGEMENT

Internet campaigns are collections of rules, optionally coupled with targeted, personalized e-mail, that work together to accomplish a specific goal. Using the Campaign Manager within the Personalization workspace, the business user can create new campaigns by entering the campaign name, description, and start and stop dates and times. This information can be edited at any time.

After creating the campaign, you're provided with a list of content spots within the project and can choose the content spots whose default rule mappings should be overridden. For example, to create the "Holiday Specials" campaign, you might want to change the "SalesSpot" content spot to use the "Get Holiday Sale" rule instead of the "Show Nothing on Sale" rule. To do this, select that content spot to be shown a list of all the rules that include sale items, and choose the "Get Holiday Sale" rule. When the campaign is published and the corresponding start date occurs, site visitors see the sale items returned by the "Get Holiday Sales" rule. Within a campaign, you can override as many or as few content spot-to-rule mappings as desired.

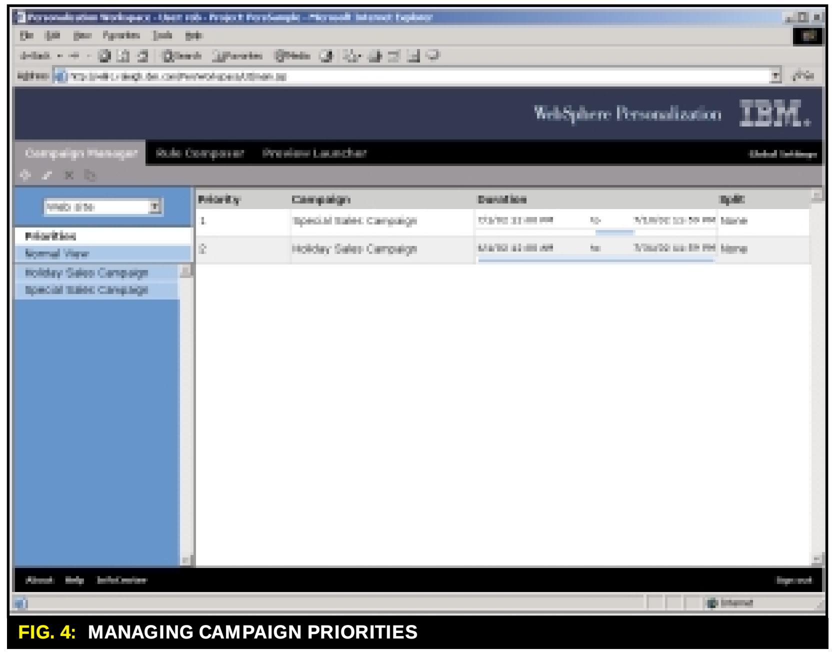
Multiple campaigns can also be defined. When multiple campaigns are defined, they're often for different, nonoverlapping periods of time, or if the time periods overlap, they typically override different content spot–to–rule mappings. When time and content spots overlap, the mapping from the campaign with the highest priority is used. You can manage campaign priorities from within the Campaign Manager as shown in Figure 4.

In addition, the campaign rule mappings feature can be used to "split" a single content spot to randomly map to two or more rules. For example, to see how a different rule might work on a given spot, you could create a campaign and for a period of one week have half the site traffic use the original rule (the "champion") and half the site traffic use the new rule (the "challenger"). After the week is over, using the rule effectiveness reports within the WebSphere Site Analyzer component, you can determine which rule was most effective and either replace the champion with the challenger or just discard the challenger.

PREVIEW

In addition to creating and editing rules within the workspace, you can easily see how the site will look for different users at different times.

Using the Preview Launcher, pseudo-users (named users that exist only in the preview system) can



Priority	Campaign	Duration			Split
1	Special Sales Campaign	[illegible]	to	[illegible]	None
2	Holiday Sales Campaign	[illegible]	to	[illegible]	None

FIG. 4: MANAGING CAMPAIGN PRIORITIES

quickly and easily be created. There's no need to set up a database or LDAP repository. Simply hit the add button and type in the new pseudo-user's properties based on the site's defined user profile. For example, you might create a user who's a manager from Seattle. After a few pseudo-users have been created, creating additional pseudo-users is even easier. If you're previewing the site as a manager from Seattle and want to see what the site would look like for a nonmanager from Raleigh, simply hit the copy button to copy the manager from Seattle and then change the properties as shown in Figure 5.

After creating one or more pseudo-users, use the preview button to specify a date and time, or accept the current time, and to view the site as it will appear for a user with the specified attributes at the specified date and time. While in preview mode, each page is decorated with small blue circles wherever a content spot has invoked a rule to produce output, as shown in Figure 6.

If you don't like the results or want to understand why certain content is being produced, hover the mouse over the spot to see the rule name, or click on it to go to a dialog. The dialog will allow you to change the content spot–to–rule mapping (without going to the full Campaign Manager), or to go to the Rule Editor for the rule being used. This shortcut allows you to quickly and easily get the site running in the desired manner.

IMPLICIT PROFILING

The Personalization component makes it easier to gather information about site visitors and to act on that information in real time using the category object. Page designers and servlet writers can instrument their pages and code to record the categories of information the user is viewing or dealing with. For example, if the user is looking at an article about football, the JSP that displays the article can easily record this information in the category object.

The category object is an implicit object in the Rule Editor, which means it's always present. Rules can use this category information to make decisions like "if categories.totalfootball views > 10, then ..."

The category information lasts only during the current session. To make use of the knowledge gained on subsequent sessions, create an update rule to update the user profile with the newly gained information, either by incrementing a corresponding count within the user profile, or by storing summary information.

The site defines category names, and naming is hierarchical. For example, Football/Giants is a different team than Baseball/Giants. The hierarchy notation is more than just a longer name; the intermediate fields have meaning. If a user looked at an article tagged with a category of Football/Giants, and another tagged with Football/Broncos, then the count of Football articles would be two, while the count of Football/Giants and Football/Broncos would be one.

Finally, the information gathered within the category object is optionally stored in the Site Analyzer's log for offline reporting and analysis.

CAMPAIGN AND RULE EFFECTIVENESS

When a rule fires, the name of the campaign, the name of the rule, and the list of resources returned are optionally logged to the Site Analyzer database. Later, JSPs or servlets that process subsequent events (such as click-through on an ad or drill-down for more information on an article) can call an API to record the event (the resource ID and a success or failure code). Site Analyzer can then write reports that analyze the effectiveness of each campaign and/or rule being used.

RECOMMENDATION ENGINE

The Personalization component includes the LikeMinds Personalization Server (LPS). LPS provides Java APIs to five recommendation engines.

- ***Preference engine:*** This engine generates recommendations using collaborative filtering algorithms based on users' item ratings. This is the engine used in the sample application (Movie Critic).
- ***Purchase engine:*** This engine, which accesses transaction information, generates recommendations based on purchase data, such as which items were purchased, how many times, when, and so on. Transaction databases typically have large amounts of data that can be used to seed a new LPS installation. Thus, purchase data from a transaction database allows LPS to begin generating recommendations right away.
- ***Clickstream engine:*** This engine, which also accesses transaction information, generates recommendations based on users' shopping behavior as they navigate a Web site (the history of user "clicks" during Web site visits) and the items users view, click on, and add to their shopping carts.
- ***Product-matching engine:*** This engine works with the preference engine, purchase engine, or clickstream engine. It expands the recommendations for each user by adding items similar to the items that have already been recommended. The similarity of items is determined by the item attributes configured for the product-matching engine. The product-matching engine can make recommendations for items that have just been added to the database.
- ***Market-basket analyzer:*** This engine provides up-sell and cross-sell recommendations and uses a different technique (that is, not collaborative filtering).

Resources

The resource engine provides an abstraction layer allowing user and content to be defined to the Personalization system in a consistent manner. The major concepts of the resource engine are resource collections, resources, and resource instances. A resource is a Java class that defines the properties of the user or content object. In database terms, it's analogous to the database schema that defines the column names and types for a database table. A resource instance is an instance of the resource class. Again, using a database analogy, the resource instance is like a row within a database – it contains actual values for each property defined by the resource. Finally, a resource collection is a collection of resource instances and is like a database table (with a fixed schema and a number of rows).

While resource collections, resources, and resource instances are easily mapped to familiar database concepts, the actual content store that they wrap doesn't have to be a database table. It can be a file system, an LDAP repository, an XML store, or virtually any content store accessible by Java.

You define a resource collection by creating two or three Java classes and registering those classes to the resource engine under a given resource collection name. The first class is the resource class with a specific set of properties. The properties can be any of the common primitive Java types (string, int, float, date, etc.), an array of one of the primitive types, or an array of Java objects. The second class contains some life-cycle methods (create, delete, sync [update]) and a set of find methods (findbyID, findbyQueryString, etc.). This second class can be split into two classes.

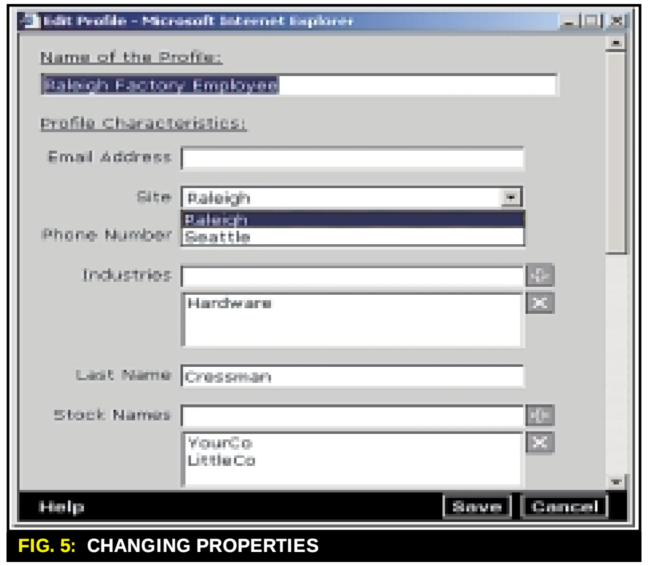
When the resource datastore is a database, an LDAP directory, or an EIP-accessed data store, then the resource implementation classes are most easily created using the Resource Definition wizards in Studio Advanced. These wizards can even generate the complex code necessary to retrieve and search on multivalue properties stored within a set of database tables. If some other type of datastore is being used, the resource implementation classes are created by a programmer using WSAD, WebSphere Studio Site Developer, or a similar Java IDE.

In addition to resource collections, virtually any Java objects can be defined for use within a rule. Such objects are defined to the resource engine as application objects – application objects are defined to the resource engine by specifying a name and the name of the class that implements the application object. The application then stores a reference to the application object within the HTTP session using the same name. In this way, the properties of the object are available to the business user for creating rules.

Conclusion

The Personalization component of WebSphere Portal extends the usual programming environment for your portal and enables you to change the targeting decisions within a portlet to react to business needs.

When coupled with a third-party Web content management solution, or with the Web content publishing capabilities of WebSphere Portal, personalization can be even more powerful because business users can create the content and targeting strategies quickly and easily.

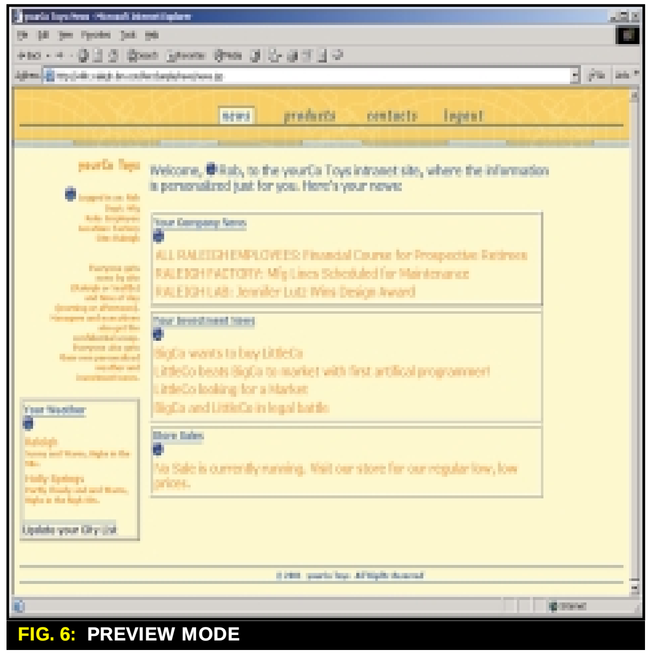


Name of the Profile:
Raleigh Factory Employee

Profile Characteristics:

Email Address	
Site	Raleigh Raleigh Seattle
Phone Number	
Industries	Hardware
Last Name	Cressman
Stock Names	YourCo LittleCo

FIG. 5: CHANGING PROPERTIES



Welcome, Rob, to the yourCo Toys intranet site, where the information is personalized just for you. Here's your news:

Your Company News
ALL RALEIGH EMPLOYEES: Financial Course for Prospective Retirees
RALEIGH FACTORY: Mfg Lines Scheduled for Maintenance
RALEIGH LAB: Jennifer Lutz Wins Design Award

Your Investment News
BigCo wants to buy LittleCo
LittleCo beats BigCo to market with first artifical programmer!
LittleCo looking for a Market
BigCo and LittleCo in legal battle

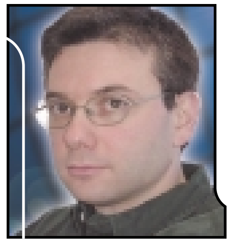
Store Sales
No Sale is currently running. Visit our store for our regular low, low prices.

FIG. 6: PREVIEW MODE

Discovering and invoking the Web service

Developing Web Services with WebSphere Studio PART 2

BY RON BEN-NATAN



ABOUT THE AUTHOR
Ron Ben-Natan, CTO of ViryaNet Inc., holds a PhD in computer science in the field of distributed computing and has been architecting and developing distributed applications for over 15 years. Ron's hobby is writing about how technology is used to solve real problems and he has authored numerous books, including *IBM WebSphere Application Server: The Complete Reference*, published by Osborne/McGraw-Hill.

E-MAIL
rbennata@hotmail.com

In my last article (*WSDJ*, Vol. 1, issue 4) I showed you how to use WebSphere Studio Application Developer (WSAD) to develop and publish a Web service. You saw how to use the Web services wizard to wrap an existing Java method as a Web service and expose the metadata required for invoking the service. You also saw how the UDDI Explorer is used to publish your service on a public registry so others can find and use it. This month's focus is on discovering the service and building a client that invokes the Web service. You'll learn more about how WSAD hides the complexity and mechanics of Web services by introducing a set of tools and wizards. You can do all your Web services development quickly and efficiently, without writing a single line of XML.

Last time I focused on the Web service developer (and publisher); this month I focus on the consumer – the developer of the client application that makes use of the Web service. WSAD supports not only Web service development but also Web service consumption, all using the same set of tools hiding the complex mechanics of the Web service.

Continuing the scenario introduced before, recall that the Web service that was developed used advanced optimization algorithms to create efficient driving routes. People who need to visit a set of locations during a workday access the service by passing in a set of latitude/longitude pairs that need to be visited. The Web service uses a smart routing and optimization engine and has access to historical traffic patterns with which it can suggest an optimal sequence to minimize driving time. The service can be used by applications for delivery trucks, service technicians, claims adjustors, and more.

The Java code that performed this optimization had a method with the following signature:

```
public int[]
getWorkOrderSequenceForEmploy
ee(long[] lats, long[] longs)
```

It receives as input two arrays with the latitude and longitude coordinates for the places to be visited. The reply is an array with the sequence. For example, if it has passed 10 locations it may return an array of the form {4,6,2,3,1,8,9,7,0,5}, meaning the person should first visit the location with the fourth latitude/longitude, then the location with the sixth latitude/longitude, and so on. This was wrapped as a Web service and published on a UDDI registry. Let's move on to see how a client application developer becomes a consumer of this Web service.

Discovering and Importing the Service

As the Web service developer, you published the work order sequencing service on IBM's Test Business UDDI site – a public registry deployed on the Web. This month you're switching roles – you're now the consumer of the Web service. WSAD also supports this role.

As a Web consumer you want to be able to search the UDDI repository and find the business and service that will help you develop your application. Once you find it, you want to easily make use of the published service. WSAD helps you do both using a set of tools – you don't have to make complex UDDI API calls or hard-to-write SOAP messages.

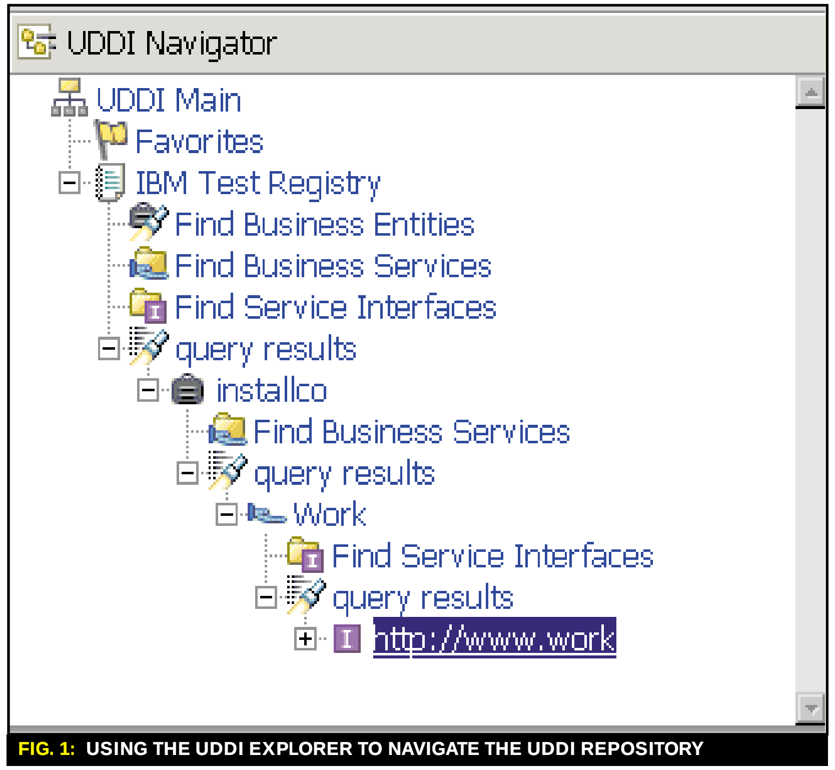
The UDDI Explorer is used to discover the service. This is the same tool used for publishing the service. Now you need to use the import mode. The UDDI Explorer is a visual tool that allows you to navigate a UDDI repository instead of making UDDI API calls to search the repository.

To import a Web service use File –>Import from the menu bar, then select the UDDI option from the wizard. This brings up the UDDI Explorer. You use the UDDI Navigator – the left pane in the UDDI Explorer – to search for a business based on the classification categories you're interested in. Within the business entity, you can search for the business services offered by this business (see Figure 1). The final element you need to select is the actual service definition. This is the URL that contains the WSDL document defining the service. At this point you can go ahead and click the Import to Workbench tool highlighted in blue on the Actions toolbar of the UDDI Explorer (see Figure 2). You've now completed the first part of the consumption process – discovering the service and importing the WSDL file. Your project within WSAD will include the WSDL file you published last month.

Creating the Proxy and Running the Sample Test Application

Now that you have the WSDL you can see where the service implementation resides, what it assumes as input, and what kind of output it generates. You can build SOAP documents and send them over HTTP to be serviced by the last column's implementation . . . but should you? Not really. Once more, WSAD and the set of Web services tools make things simpler. Using the Web services wizard you can create a proxy that works on your behalf for generating the SOAP calls to the Web service.

Before looking at the wizard and the way WSAD generates the proxy, let's look at the proxy's role and the life cycle of an invocation. The proxy class is generated on your behalf by the Web services wizard based on the WSDL file. As its name suggests, it's a surrogate that lives in your program's address space, i.e., it's a local Java object. It has a method that conforms to the signature as defined by the WSDL file. Since the WSDL definition is based on the original method's signature, it mimics the signature of the method you originally wrapped as a Web service. When you invoke the proxy's method it generates the SOAP message based on the definitions within the WSDL and forwards this SOAP message to the endpoint as defined in the WSDL. On the server side, the rpcrouter (part of the Web services infrastructure) unpacks the SOAP message, converts the XML to the Java objects forming the input arguments, and makes the call on the object implementation. The method generates a return value that's converted by the rpcrouter to XML and returned to the proxy. Back on the client, the proxy converts the XML back to Java objects based on the encoding type. Figure 3



UDDI Navigator
- UDDI Main
  - Favorites
  - IBM Test Registry
    - Find Business Entities
    - Find Business Services
    - Find Service Interfaces
    - query results
      - installco
        - Find Business Services
        - query results
          - Work
            - Find Service Interfaces
            - query results
              - http://www.work

FIG. 1: USING THE UDDI EXPLORER TO NAVIGATE THE UDDI REPOSITORY

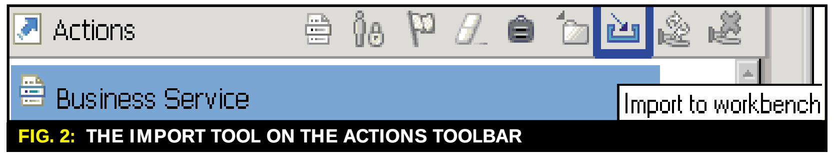


FIG. 2: THE IMPORT TOOL ON THE ACTIONS TOOLBAR

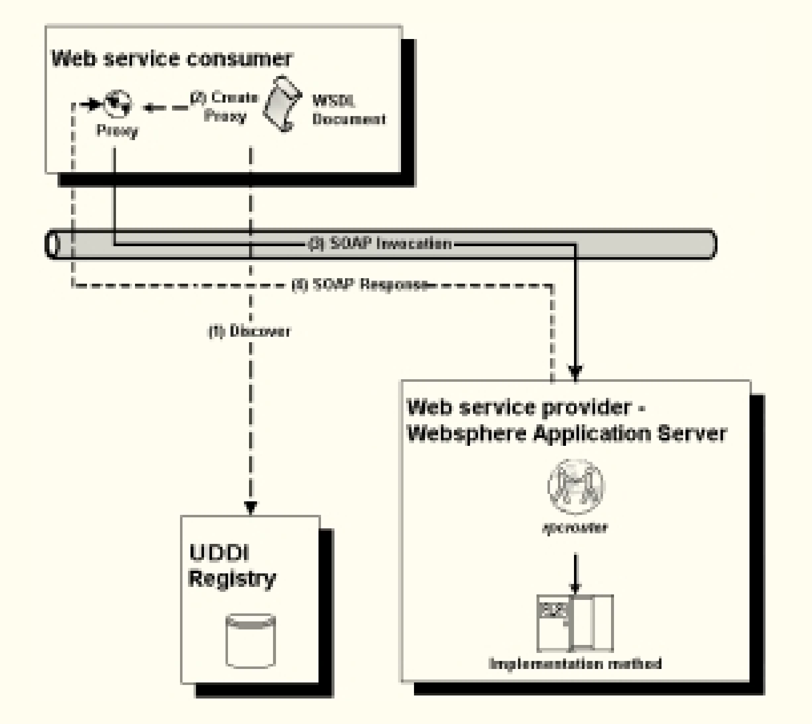


FIG. 3: INVOKING A WEB SERVICE THROUGH THE PROXY

demonstrates this process.

Use the Web services wizard to import the Web service and generate the proxy. The wizard allows you to create a Web services client as shown in Figure 4. It takes you through a series of steps, asking simple questions such as whether you want to generate a sample application that uses the proxy, whether you want to launch the sample, etc. When you click the Finish button on the wizard, it goes to work and generates all the plumbing for you. Once more, you don't have to write a single line of code to be able to consume the Web service. It not only generates a proxy Java class, it generates a set of JSPs that use the proxy as a way for you to test the Web service (and, of course, to copy/paste from to build your own consuming application). It's really that simple.

A Peek Behind the Scenes

The plumbing generated by the Web services wizard is interesting. For every class you select in the wizard, a proxy class is created. An example is shown in Figure 5. For every method that serves a proxy method to a Web service, WSAD generates a method similar to that shown in Listing 1 (some of the details are omitted for the sake of brevity). As you can see, the proxy method uses a call object to package the invocation details. The call object is part of the underlying infrastructure within the Apache Web services package (used in the WebSphere Application Server as well as in WSAD).

Venturing into the Unknown

Let's live on the wild side. Web services are about interoperability and distributed systems, right? Let's take a brief look at what happens when you want to publish your Web services for use by a .NET client. It's actually quite simple. Visual Studio .NET has an import program similar to the Web services wizard in WSAD that takes the WSDL definition and creates a proxy that takes care of the plumbing. You invoke it using a command of the form

```
wsdl.exe /language:VB <URI-
for-WSDL>
```

if you want to generate a proxy in VB, or a command of the form

```
wsdl.exe  <URI-for-WSDL>
```

if you want the proxy in C#. A small part of the generated VB proxy is shown in Listing 2, and a small part of the C# proxy is shown in Listing 3. The proxy functions much like the one generated by WSAD. One interesting difference is that the proxy generated by WSAD uses a delegation model using an embedded call object while VS.NET proxies use an inheritance model – these proxies are subclasses of the SoapHttpClientProtocol class where all the magic takes place. Therefore, the invocation is through the Me object in VB and through the this object in C#.

Summary

This column completes an introduction to using WSAD to develop Web services. The two-part series walked you through the fundamentals of developing and publishing a Web service using WSAD as well as discovering and invoking the Web service using the support infrastructure provided by WSAD and by WAS. What is important to take from this series is that the WebSphere tools make all this easy, and you don't have to know any of the low-level details to participate in this brave new world.

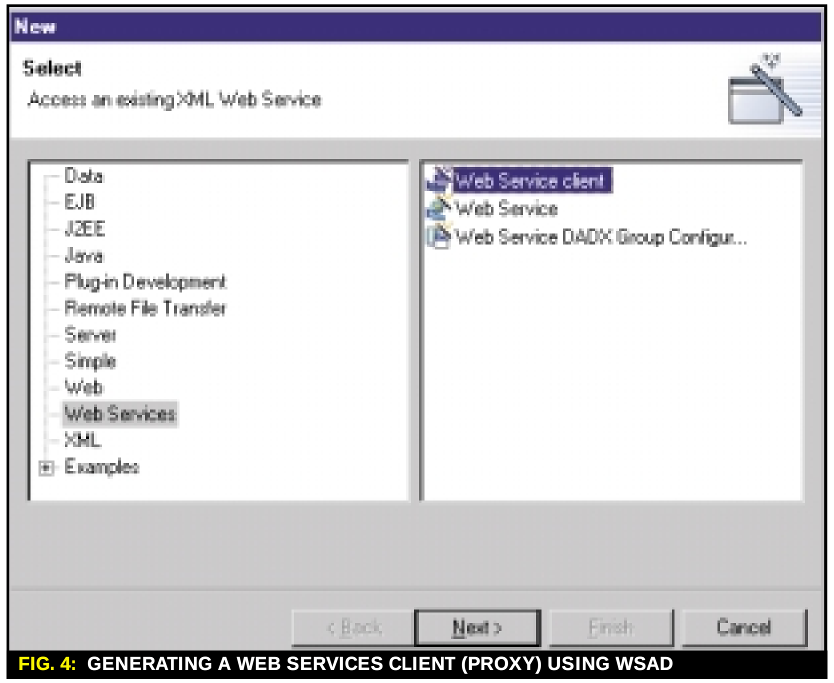
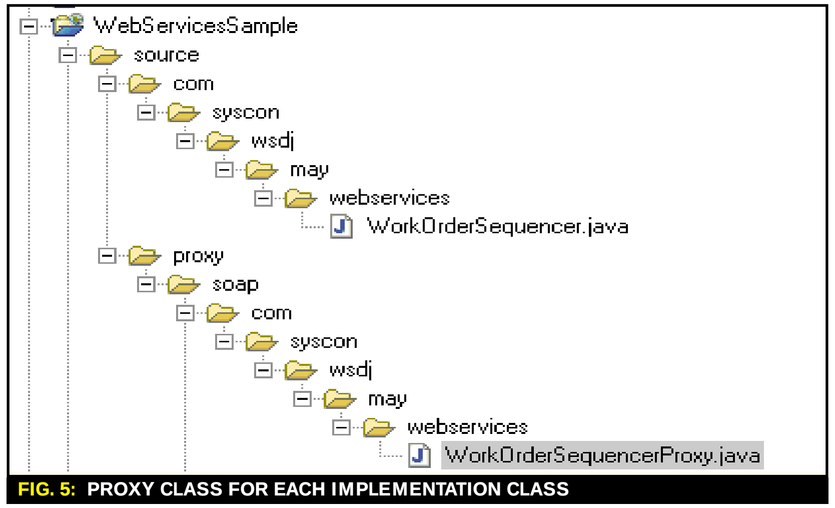


FIG. 4: GENERATING A WEB SERVICES CLIENT (PROXY) USING WSAD



```
WebServicesSample
  source
    com
      syscon
        wsdj
          may
            webservices
              WorkOrderSequencer.java
  proxy
    soap
      com
        syscon
          wsdj
            may
              webservices
                WorkOrderSequencerProxy.java
```

FIG. 5: PROXY CLASS FOR EACH IMPLEMENTATION CLASS

LISTING 1: WSAD GENERATES A PROXY METHOD

```
public synchronized int[]
getWorkOrderSequenceForEmployee(
        long[] lats,long[] longs) throws Exception
 {
  String targetObjectURI = "http://..";
  String SOAPActionURI = "";

  if(getURL() == null)
  {
   throw new SOAPException(Constants.FAULT_CODE_CLIENT,
    "A URL must be specified ...");
  }


call.setMethodName("getWorkOrderSequenceForEmployee");
  call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
  call.setTargetObjectURI(targetObjectURI);
  Vector params = new Vector();
Parameter latsParam = new Parameter(
   "lats", long[].class, lats,
Constants.NS_URI_SOAP_ENC);
  params.addElement(latsParam);
Parameter longsParam = new Parameter(
   "longs", long[].class, longs,
Constants.NS_URI_SOAP_ENC);
  params.addElement(longsParam);
  call.setParams(params);
  Response resp = call.invoke(getURL(), SOAPActionURI);

  //Check the response.
  if (resp.generatedFault())
  {
   Fault fault = resp.getFault();
   call.setFullTargetObjectURI(targetObjectURI);
   throw new SOAPException(
       fault.getFaultCode(), fault.getFaultString());
  }
  else
  {
   Parameter refValue = resp.getReturnValue();
   return ((int[])refValue.getValue());
  }
 }
```

LISTING 2: GENERATED VB PROXY

```
'-------------------------------------------------------

' <autogenerated>
'     This code was generated by a tool.
'     Runtime Version: 1.0.3328.4
'
'     Changes to this file may cause incorrect behav-
ior          and will be lost if
'     the code is regenerated.
' </autogenerated>
'-------------------------------------------------------


Option Strict Off
Option Explicit On
Imports System
Imports System.ComponentModel
Imports System.Diagnostics
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Xml.Serialization

'
'This source code was auto-generated by wsdl,
Version=1.0.3328.4.
'
'<remarks/>
<System.Diagnostics.DebuggerStepThroughAttribute(),  _

System.ComponentModel.DesignerCategoryAttribute("code"),
_
 System.Web.Services.WebServiceBindingAttribute(Name:=
"WorkOrderSequencer",
[Namespace]:="http://tempuri.org/")>  _
Public Class WorkOrderSequencer
    Inherits
System.Web.Services.Protocols.SoapHttpClientProtocol

    '<remarks/>
    Public Sub New()
        MyBase.New
        Me.Url = "http://localhost:8180/…"
    End Sub

    …
    Public Function getWorkOrderSequenceForEmployee …
        Dim results() As Object =
Me.Invoke("getWorkOrderSequenceForEmployee", New
Object() {…})
        Return …
    End Function
    …
End Class
```

LISTING 3: C# PROXY

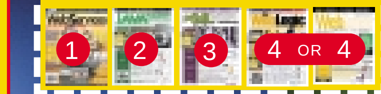
```
//-------------------------------------------------------

// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version: 1.0.3328.4
//
//     Changes to this file may cause incorrect
        behavior and will be lost if
//     the code is regenerated.
// </autogenerated>
//-------------------------------------------------------


using System.Diagnostics;
using System.Xml.Serialization;
using System;
using System.Web.Services.Protocols;
using System.ComponentModel;
using System.Web.Services;


/// <remarks/>
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")
]
[System.Web.Services.WebServiceBindingAttribute
  (Name="WorkOrderSequencer", Namespace="http://tem-
puri.org/")]
public class WorkOrderSequencer:
System.Web.Services.Protocols.SoapHttpClientProtocol {

    /// <remarks/>
    public WorkOrderSequencer () {
        this.Url = "://localhost:8180/…";
    }
    …
    public … getWorkOrderSequenceForEmployee (…) {
        object[] results =
this.Invoke("getWorkOrderSequenceForEmployee ", new
object[] {
                        …});
        return …
    }
    …
}
```

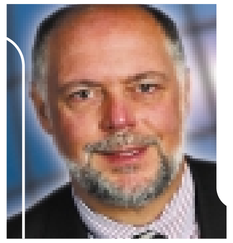


Strategies for information access and delivery in the WebSphere Portal environment

The Business Intelligence Imperative

BY GARY GOLDBERG

Business intelligence (BI) is one of the most vibrant sectors of the software industry, with many research firms projecting double-digit growth in the years ahead. The reason is simple: more people than ever before are demanding information. BI software allows users to find the information they need quickly so they can make better decisions.



ABOUT THE AUTHOR

Gary Goldberg is general manager of WebFOCUS applications at New York–based Information Builders, Inc. He is currently responsible for developing the WebFOCUS portal integration products and integrating WebFOCUS with vertical applications such as PeopleSoft and SAP.

E-MAIL

gary_goldberg@ibi.com

As enterprise information portals (EIPs) rise to the top of the CIO agenda, it's important to understand the ground rules for a successful business intelligence strategy. Informed business decisions aren't based on any single set of criteria – they draw on information from many different sources. Most intranets began as repositories of static information, such as company news and HR data. But these sites have the potential to become much more. EIPs are dynamic and interactive. They can include triggers that alert users when new reports have been posted, and logic that kicks off events in enterprise applications. Some portals even let users conduct transactions such as procurement of office supplies.

In order to successfully integrate BI technology into an EIP environment, there are several major issues that must be addressed.

- The ability to access a broad set of data sources on various computing platforms
- The need for a highly granular solution based on a component-based development architecture
- The ability to scale portal assets to serve a large, geographically diverse user population
- The need for personalized access to information.

This article describes an effective strategy for integrating BI technology within the IBM WebSphere Portal environment. It examines the basic BI requirements for building effective EIPs, using Information Builders' WebFOCUS BI software as an example of how developers can meet today's corporate EIP requirements.

WebFOCUS Open Portal Services enable WebSphere users to add advanced data access and information delivery capabilities to their portal environments. This functionality is achieved through a set of reusable EIP components built as JSPs and easily integrated as portlets into existing WebSphere Portal environments.

The EIP Landscape

EIPs combine a diversity of Web content through a single access point by relying on technologies that users are comfortable with: browsers and search capabilities. What differentiates portals from static Web sites is their ability to incorporate data from multiple sources in multiple formats and organize it in a consistent way. Just as users can jump from news reports to shopping malls on Yahoo, EIPs enable them to jump from business intelligence systems to ERP applications to sales reports, all without leaving their browsers.

An EIP provides a focal point from which to access many disparate types of business information. End users have different skill sets, and each user requires unique content and levels of detail. Thus, a successful BI solution must be able to provide a uniform data-access, integration, and reporting environment that satisfies all the information needs of the enterprise, regardless of individual requirements and skill levels. Users should be able to personalize the reporting interface so they see only the information that's important to them.

In short, the portal's job is to concentrate information from multiple sources into a single location on the Web. Often that information is locked inside databases and applications. The ability to access multiple disparate data sources is one of the critical success factors in an EIP strategy.

Data Access and Integration

While portals bring exciting new capabilities to users, the ground rules haven't changed for IT professionals. Portals are merely a new mechanism for delivery and presentation of information to a diverse user population. As with any IT initiative, the heart of the matter involves accessing, analyzing, and distributing information.

I like to use the analogy of an hourglass turned on its side. At one end are all the data sources, spawned by a wide range of production systems. The data must be funneled through a narrow focal point at the middle of the hourglass, where it's consolidated, merged, and cleansed. As it's funneled out the other end, it's delivered to a wide range of people with a wide range of needs. An EIP can support multiple views of the same information for many different users and deliver consistent answers, no matter how the data is summarized.

A typical enterprise depends on many data sources, including relational database management systems; legacy DBMS systems and file systems; CICS and IMS/TM transaction systems; MQSeries and MQSeries Integrator-based messaging applications; SAP, PeopleSoft, and J.D. Edwards application environments; and custom-developed applications written in COBOL, RPG, C, C++, and other callable languages. Much of the information portal users need is locked up in these production databases and file systems.

This is one reason why portals need to be able to interact with BI tools to do their jobs. BI tools can unlock this data and make it easily deployable through the portal – if they have the right back-end infrastructure for data access and conversion. It's important to select a business intelligence environment that can natively access many types of data.

Customers such as Beringer Vineyards, NASA, and Virginia Commonwealth University are using WebFOCUS to consolidate data and publish reports generated by URLs linked by an EIP. These customers were drawn to the Web-based capabilities for information access and delivery, the pivot point for accessing both real-time and historical information from internal applications, legacy databases, or the Internet at large. WebFOCUS provides real-time data access and integration of more than 85 information sources – including legacy, relational, and ERP data – across more than 35 platforms. This broad back-end access helps customers integrate the information they need, when they need it.

Granularity Through Portlets

To properly deploy BI systems in the portal space, reports and BI routines need to be broken into components. In the case of WebSphere, the components are called portlets.

WebFOCUS Open Portal Services for IBM WebSphere Portal allows customers to add business intelligence content to their portals without programming. A series of portlets, designed specifically for the WebSphere environment, lets users personalize the way they view, store, and retrieve business intelligence content. For example, they can decide what content appears in each section, how that content is displayed, and how it's organized. System administrators can assign access privileges to users at any time, allowing them to be more self-sufficient as company needs change.

Most BI environments lack this level of integration. Rather than executing native portlets to perform designated reporting and analysis tasks, the two environments are executed separately. Users temporarily leave the portal environment to interact with the BI tool, then return to the portal for other activities. While many reporting products make information available in this way, it's not the ideal approach since it prevents true integration of BI capabilities into the portal environment.

By utilizing the inherent flexibility of WebSphere Portal, WebFOCUS Open Portal Services simplifies the user experience. Portlets can be called to launch reports, create lists, navigate reports, and perform other tasks. Some are fixed portlets that developers define. Others allow users to define the content they need.

The WebFOCUS portlets are grouped into Administration Services, Content Display Services, and Content Navigation Services:

- ***Administration Services:***
  - –*Security interface:* Controls login options, including silent logins and separate logins.
  - –*Report status:* Allows users to access a status window for deferred report execution.
- ***Content Display Services:***
  - –*Display fixed report/list:* Allows administrators to set the content of the window to specific reports or lists.
  - –*Display personal report/list:* Enables users to personalize the portlet, for example, to select a report or list of reports to display.
  - –*Launch WebFOCUS products or tools:* Used to bring up a separate WebFOCUS product such as the WebFOCUS Report Assist ad hoc tool.
- ***Content Navigation Services:***
  - –*Role Tree:* Allows the user to navigate a report tree based on his or her role in the security scheme. This tree includes content from all the domains the user can access.
  - –*Domain Tree:* Allows the user to see the content in a given domain (subject area) and to move from one domain to another.
  - –*Search Tree:* Generates a search interface for finding reports within the WebFOCUS environment. (An example of WebFOCUS Portlets integrated with WebSphere. This view shows the deployment of all three types of portlets – administration, display, and navigation.)

Personalized Access to Information

By delivering information from more sources, more efficiently, in a more usable format, BI tools enhance the value of the portal. The decision-making process throughout the organization is also enhanced, resulting in companies that are more nimble, productive, and competitive. By tailoring an EIP to different skill levels, developers can ensure all users have the tools to perform their jobs.

User preferences are a factor in any portal strategy. People want to see information in a way that makes the most sense to them. To be effective, information must also be relevant to individual jobs and functions. The EIP should provide a flexible view of the information for many different types of users inside and outside the organization. This flexibility allows access to information to be personalized according to skill level, need, and privileges.

Does one audience use Excel? Does another prefer e-mail? Are they viewing their information from desktop PCs or PDAs? Do they think in terms of words or pictures, or do they use specialized analytical tools? Do they like to research information, or do they prefer to have it delivered to them?

When it comes to building an EIP system, you need to select one technology that can deliver information in many different ways, supporting a variety of formats and output devices. The maintenance and integration implications of using a different technology for each audience are major cost factors.

Flexible Report Distribution

To add value beyond delivering content to the portal on demand within the EIP context, a BI environment must offer mechanisms for automating and scheduling the distribution of reports to the portal, as well as via e-mail, printers, fax, or mobile devices, to any user in any format, including HTML, DHTML, XML, Excel, and PDF.

With WebFOCUS Report Caster and WebFOCUS Open Portal Services, reports can be scheduled daily, weekly, monthly, or at any other time interval. Based upon predetermined events, system administrators can set triggers that will automatically run and distribute reports. This enables the portal to notify people when something significant occurs – for example, when a big customer cancels an order or the marketing department goes over budget.

WebFOCUS administrators can also control the timing and frequency of report generation and distribution. Support is provided for both one-time and recurring reports. A subscription function lets recipients bypass the scheduling feature and subscribe to reports on their own. Additionally, administrators can run individual reports and burst the results into multiple sections, with each section then sent to a different Web server or e-mail address. WebFOCUS can burst reports according to any number of sort variables, including name, branch office, sales region, department, or management level.

Flexible Deployment Architecture

An effective BI architecture should employ a multitier architecture to improve load balancing, minimize distribution of application components, and maximize network performance. This way, data access services can be deployed on one platform, business logic on another, and presentation logic on a third. All queries can be dynamically partitioned, guaranteeing that the complex number-crunching and aggregation will take place on the back-end database server, not the Web server or desktop. The answer set, in the form of an HTML page, can then be sent back over the network and viewed in the user's browser, minimizing network traffic. This architecture offers maximum scalability and usability and supports multiple security implementations.

WebFOCUS employs a true thin-client architecture with only HTML delivered to the browser. The components are installed wherever the user requires. The data access components are stored near where the database resides for efficient real-time reporting. The balance of the application resides with WebSphere under the control of the application server.

Inherent Scalability

One of the differentiators of portals as a mechanism for delivering information is the ability to scale to support large volumes of users, both inside and outside the enterprise. Most portal vendors take a Java-centric approach to EIP development using J2EE. This is the right approach for the accompanying BI tools as well, since it's highly scalable and platform- neutral. The best-case scenario is to have a BI solution that can be deployed as a set of Java servlets and components. This enables the BI environment to scale along with the portal environment to virtually any size.

Additionally, an efficient BI architecture can be deployed on multiple platforms, allowing processing to be performed close to the data in order to minimize the volume of information flowing through the network. Front-end portal components should use a thin-client architecture, requiring only a browser for users to access the complete resources of the portal.

End-User Personalization

End-user personalization refers to the ability of users to personalize the way they view, store, and retrieve information. Not every user has the same information needs. And not every user wants to see information in the same way. Companies must be able to differentiate users based on each individual's need-to-know, responsibilities, and personal preferences. WebFOCUS integrates reporting components both at the individual level as a single dynamic report and at the group level as a set of associated reports with varied users. This enables users to personalize their reporting environments by deciding which reports they see and how they see them.

Administrator Control

WebFOCUS User Administration Services enable administrators to maintain tight security and control over their data while allowing users to personalize how they view, store, and retrieve reports. Portal administrators insert fixed WebFOCUS portlets throughout the enterprise portal environment. By making personal portlets available, WebFOCUS allows users to browse available reports and select them for delivery to a designated page in the portal.

WebFOCUS also gives administrators several ways to define and control user rights within the reporting environment. Administrators can maintain user IDs and passwords within the WebFOCUS environment or set WebFOCUS to integrate with external security within the WebSphere Portal. System administrators can also determine the type and quantity of information that each user can see, managing the entire reporting environment from one integrated console. Administrators can structure and assign reports to individuals based upon the department they work in and can assign even greater levels of security by granting access to reports based upon the underlying data values or the user's role in the organization.

Departmental security allows users to see only reports for the departments to which they have access, such as accounting, finance, or sales. Role-based security allows individuals to view reports based upon their particular roles within the organization, for example, manager, executive, or line worker.

Conclusion

Succeeding in business depends on how well you know your customers, how well you understand your business processes, and how effectively you run your operations. Unless you can directly access and integrate your data, transform it into meaningful information, and then deliver that information in the right way to the right people – on demand – you won't have sufficient business intelligence to achieve business excellence. That's why portals are so critical: they give aggressive companies the edge they need to be more nimble and respond more quickly to changing market conditions.

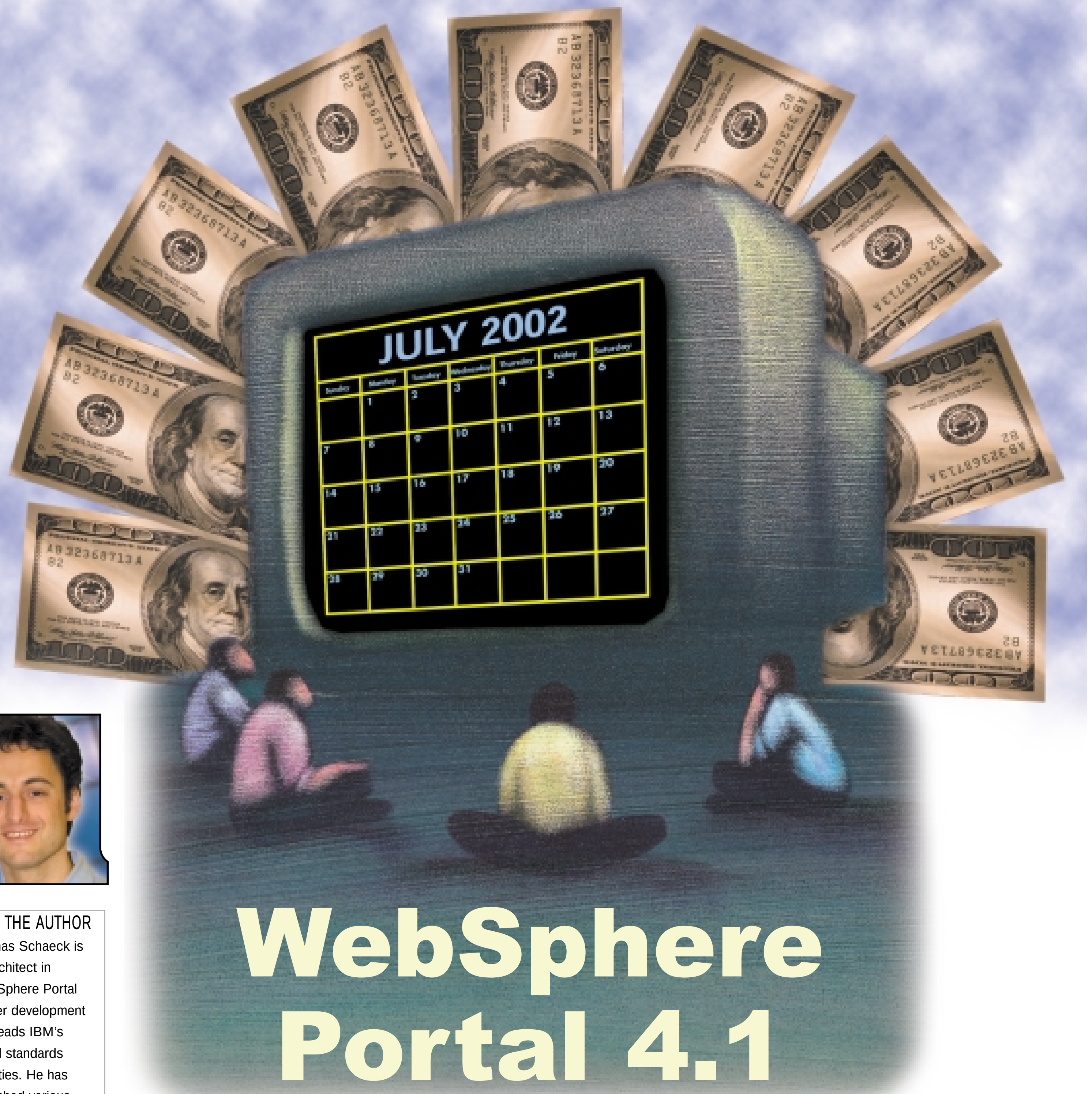
As part of an overall portal strategy, WebFOCUS Report Caster and WebFOCUS Open Portal Services supply the essential elements of successful portal deployments – universal data connectivity, personalized access to information sources, and intelligent information delivery that meets the varied needs of employees, partners, and customers. The combined capabilities of WebSphere and WebFOCUS provide customers with greater scalability and deployment for their real-time information delivery applications, delivering consistent, accurate, real-time information throughout the enterprise.

A Wish List for Developers

EIPs provide a single user interface that integrates and manages information in a variety of formats, including static HTML pages, PDF files, and spreadsheets. They also incorporate links to external applications that are relevant to the individual user's needs and skill levels. The portal thus offers a consistent, accurate, and reliable integration point for information sources to anyone who needs it.

Before embarking on an EIP strategy, organizations need to carefully consider several critical issues of portal design. How well these factors are addressed will determine how robust, comprehensive, and usable the portal eventually is—and, ultimately, its success or failure.

- On-server deployment of business intelligence applications for more efficient request processing
- Flexible single- or multiple-server deployment options so companies can easily scale their information portals to support any number of users
- Leveraging of WebSphere's load-balancing and pooling capabilities to realize efficiencies of scale as applications are deployed across heterogeneous networks
- Adherence to all Java 2 architecture guidelines to improve performance, maximize developer skill sets, and ensure immediate compatibility and deployment
- Native access to all popular data sources so developers can quickly leverage existing infrastructure and rapidly integrate pertinent data sources
- Scheduled, event-driven, on-demand information delivery so decision makers can address business issues with the formats they use every day, such as e-mail, Web pages, Excel worksheets, wireless devices, and printable PDF files
- Security at all levels to ensure the integrity, reliability, and business continuity of data and systems



WebSphere Portal 4.1 and Web Services

Local portlets and remote portlet Web services

BY THOMAS SCHAECK AND DR. CARSTEN LEUE

ABOUT THE AUTHOR

Thomas Schaeck is an architect in WebSphere Portal Server development and leads IBM's portal standards activities. He has published various papers, filed 20 patents, and coauthored *Smart Card Application Development in Java* (Springer, 2000) and *Pervasive Computing* (Addison-Wesley, 2001).

E-MAIL

schaeck@de.ibm.com

Portals provide personalized access to information, applications, processes, and people. Typically, portals get information from local or remote data sources such as databases, transaction systems, syndicated content providers, and remote Web sites. They render and aggregate this information into composite pages to provide information in a compact and easily consumable form. Many portals include applications like e-mail, calendars, organizers, banking, and bill presentment, just to name a few.

Different rendering and selection mechanisms are required for different information and applications, but they all rely on the portal's infrastructure and operate on data or resources owned by the portal, like user profile information, persistent storage, or access to managed content. Consequently, most of today's portal implementations provide a component model that allows plugging portlets into the portal infrastructure. Typically, portlets run on the portal server, processing input data and rendering output. Figure 1 shows a typical portal page.

Often, content is displayed by local portlets running on the portal. While this is useful for portlets that are used frequently and are under high load, it's not well suited to dynamic integration of business applications and information sources into portals.

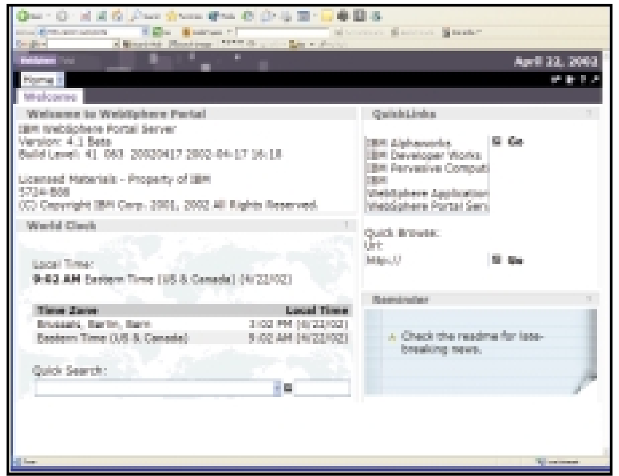
Consider the following scenario. An employee portal manager wants to include a human resources service calculating variable pay for employees and an external weather service providing weather forecasts. One solution for this scenario is depicted in Figure 2 – a human resources portlet and a weather portlet run locally on the portal server and access remote Web services to obtain the required information.

The HR portlet uses an HR Web service to calculate the variable pay. By default, it displays a form to query the required input data, for example, the employee's position. When the employee provides the data to the HR portlet, it invokes the remote Web service to calculate the variable pay based on that data. It receives the result from the Web service and displays it as a page fragment. The weather portlet by default displays weather forecasts for configurable locations and allows users to select locations in an edit mode. When the weather portlet is invoked during page aggregation, it gets the most recent forecasts for the selected locations and renders a page fragment that displays those forecasts.

This approach works only if all portlets are physically installed at the employee portal; the process of making new portlets available is tedious and expensive. To integrate HR information in the portal, either the HR department would implement the HR portlet and give it to one of the administrators of the employee portal to install, or an employee portal developer would implement the HR portlet according to the interface description of the HR Web service. Similar problems would occur for the weather portlet. In each case, significant effort is required to make the portlets available.

Obviously, it would be much more convenient if remote Web services would appear as remote portlets that include presentation and application logic, as shown in Figure 3.

Instead of just providing raw data or single business functions that require special rendering on the portal side, remote portlet Web services (RPWS) are user-facing Web services including presentation. They're aggregatable Web applications that can be invoked through a standard interface using generic portlet proxies on the portal side. No special portlet code needs to be installed on the portal. Generic portlet proxies eliminate the need for specific portlets for each Web service.



April 22, 2002

Home

Welcome

Welcome to WebSphere Portal

IBM WebSphere Portal Server
Version: 4.1 Beta
Build Level: 41_063 20020417 2002-04-17 16:18

Licensed Materials - Property of IBM
5724-B88
(C) Copyright IBM Corp. 2001, 2002 All Rights Reserved.

World Clock

Local Time:
9:02 AM Eastern Time (US & Canada) (4/22/02)

Time Zone	Local Time
Brussels, Berlin, Bern	3:02 PM (4/22/02)
Eastern Time (US & Canada)	9:02 AM (4/22/02)

Quick Search:

QuickLinks

IBM Alphaworks
IBM Developer Works
IBM Pervasive Computi
IBM
WebSphere Application
WebSphere Portal Serv

Quick Browse:
Url:
http://

Reminder

Check the readme for late-breaking news.

FIG. 1: IBM WEBSPHERE PORTAL

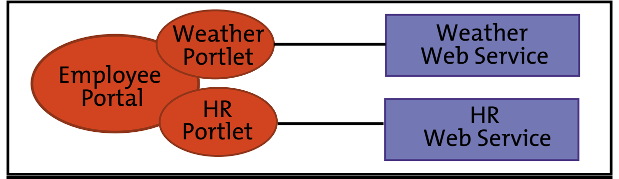


FIG. 2: LOCAL PORTLETS USING WEB SERVICES

The administrator's task is made much easier because portlets can be added dynamically to the environment, and users benefit by having more services made available to them in a timely manner. Additional remote portlets can be included in a portal. Simply find them and bind to them by creating a new portlet proxy instance bound to the RPWS. Through the use of generic portlet proxies, the RPWS appears to portals just like a local portlet and can easily be selected.

WebSphere Portal Architecture

To allow for systems using remote portlets, WebSphere Portal provides an open architecture. Figure 4 shows a high-level view of the relevant components, interfaces, and protocols.

Portal clients access the portal via the HTTP protocol, either directly or through appropriate proxies or gateways, such as WAP or voice gateways. The markup languages used by these devices may be very different. Typically, WAP phones use WML, iMode phones use cHTML, voice browsers use VoiceXML, and PC Web browsers use HTML.

When aggregating pages for portal users, the portal invokes all portlets that belong to a user's page through the Portlet API.

PORTLETS

Portlets are pluggable components running inside a portal's portlet container; they're written to a Portlet API that extends the Servlet API. However, portlets run in a portal environment while servlets run standalone in a servlet container. While servlets communicate directly with their clients, portlets are invoked indirectly via the portal application. In order to properly run in the context of a portal, portlets must produce content suited for aggregation in larger pages, i.e. portlets must produce markup fragments that can be aggregated. There are two kinds of portlets:

- ***Local Portlets:*** Run on the portlet server, local portlets are deployed by installing Portlet Archive files on portal servers and are invoked by the portal server through local

ABOUT THE AUTHOR

Carsten Leue is the chief developer for WebSphere Portal Server and the lead spec editor on the OASIS WSRP standardization committee. He has published a variety of scientific papers and is a contributor to the book *Computer Vision and Applications* (Academic Press, 2000).

E-MAIL

cleue@de.ibm.com

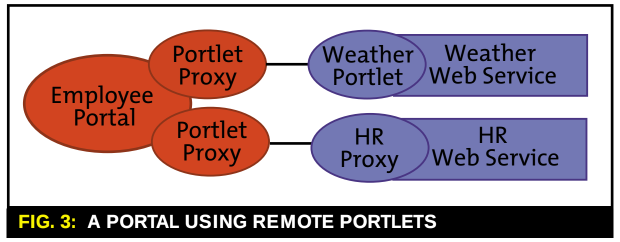


FIG. 3: A PORTAL USING REMOTE PORTLETS

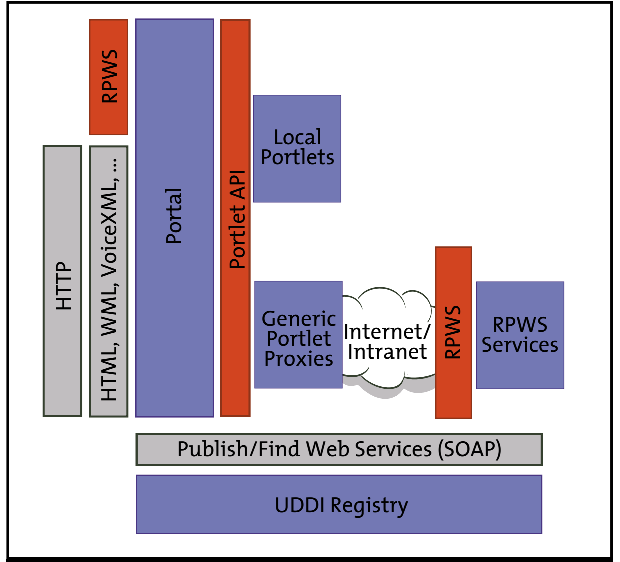


FIG. 4: WEBSPHERE PORTAL ARCHITECTURE

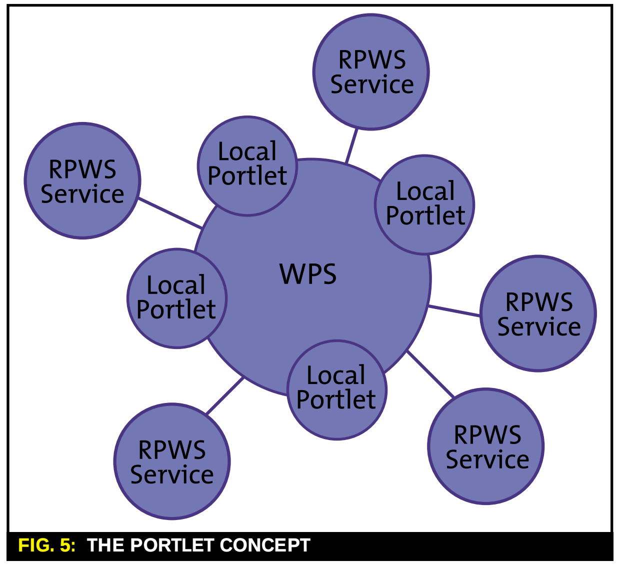


FIG. 5: THE PORTLET CONCEPT

method calls. Local portlets provide minimal latency times. However, installation usually requires assurance that the portlets are not erroneous or even malicious.

- ***Remote Portlets:*** Run as Web services on remote servers, remote portals are published as Web services in a Universal Description, Discovery, and Integration (UDDI) directory to be easy to find and bind to. An RPWS is bound by adding a portlet proxy to the portal's portlet registry when an administrator finds and selects the RPWS in the UDDI directory. Portlet proxies are generic, local placeholders that invoke portlets located on remote servers through a Remote Portlet Invocation (RPI) protocol based on the Simple Object Access Protocol (SOAP).

While local portlets can be expected to provide a large part of the base functionality for portals, the remote portlet concept allows dynamic binding of many remote portlet services without installation effort or code running locally on the portal server.

When the portal receives a servlet request, it generates and dispatches an action for the portlet affected by parameters in the request and then invokes all portlets that have to be displayed through the portlet invocation interface in a second step (see Figure 5). While portlets must implement the invocation methods required by the Portlet API, internally they may be implemented differently. A good pattern for portlet programming is Model-View-Controller, which separates the portlet functionality as follows:

* A controller receives incoming requests.
* Commands are invoked, operating on a model that encapsulates application data and logic.
* Views are called for presentation of the results.

Portlets have access to portal-related functions and data, including user profile information, persistent instance data, portlet settings, etc. Apart from portal-specific functions, portlets can use all the J2EE APIs available to servlets as well as vendor-provided connectors to access back-end data and applications or even services on the Internet.

For easier deployment, portlets can be grouped in portlet applications packaged into Portlet Archive files containing a deployment descriptor, Java classes, JAR files, and resources.

Web Services

The Web services concept allows business applications to communicate and cooperate over the Internet. Web services implies a paradigm shift from how the Internet used to work. While traditional applications interacting with services in the Internet know those services a-priori and need to be pointed to them manually, the Web services concept allows applications to find Web services in a standardized directory structure and bind to these services with minimal human interaction.

Web services allows objects to be distributed across Web sites so clients can access them via the Internet. Global service registries promote and discover distributed services. Clients needing a particular service can query the global service registry to find services that suit their needs. They can select a service, bind to it, and use it for a certain period of time. As service discovery, and selection in some cases, can be performed without human interaction, services can be switched quickly. Automated service discovery also allows robust service networks. If multiple Web services provide identical functions, clients can easily switch to a back-up system when a service fails.

The most important standards in this area are UDDI for registration and discovery of Web services; the Simple Object Access Protocol (SOAP) for communication between Web services; and the associated Web Services Description Language (WSDL) for formal description of Web service interfaces.

WEB SERVICES USED BY PORTLETS

Web services can be formally described using WSDL descriptions that can be used to generate SOAP proxies for specific programming languages. Also, there are tools that can create Web services and WSDL descriptions from existing code.

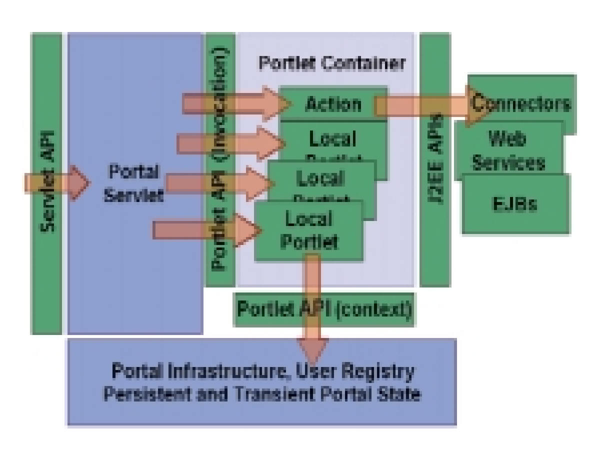


FIG. 6: WEBSPHERE PORTAL DIALOG FLOWS

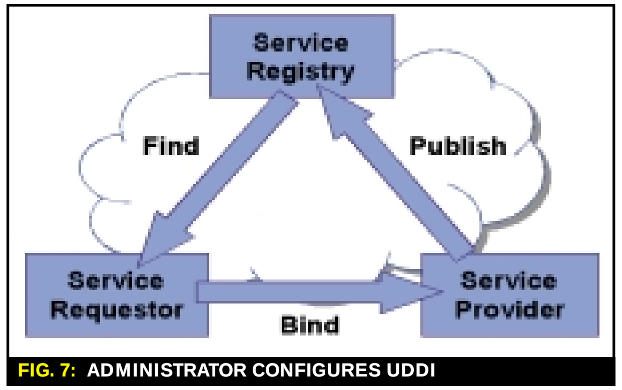


FIG. 7: ADMINISTRATOR CONFIGURES UDDI

When a portlet receives a request that requires invocation of a remote service, it makes calls on a SOAP proxy object. The proxy takes the parameters, marshals them into a programming language–independent SOAP request, and sends this request to the remote Web service. The Web ser-vice has a SOAP wrapper that receives the SOAP request, unmarshals the parameters, and invokes the local service implementation with these parameters. When the service returns the result, the SOAP wrapper marshals the result data into a programming language–independent SOAP response and sends it back to the SOAP proxy. The SOAP proxy finally unmarshals the result and returns it to the calling portlet in the form of an appropriate object.

To simplify writing portlets using Web services, IBM provides a service proxy generator tool that automatically produces client code from a WSDL interface document, and optionally a service implementation document. If only a service interface document is used, the service proxy generator tool generates a generic service proxy that can be used to access any implementation of the given service interface. If both a service interface and a service implementation are used, the service proxy generator tool generates a service proxy that will access only the specified service implementation. The service proxy contains code specific to a binding within the service interface. For example, if the binding is a SOAP binding, the service proxy will contain SOAP client code to invoke the service.

REMOTE PORTLET WEB SERVICES

To allow for dynamic integration of portlets in portals without installing portlet code locally, portlets themselves have to be provided as Web services. This requires an RPWS interface description in WSDL, which defines a common set of methods for all remote portlets and the required parameters, as well as the return values, corresponding to the Portlet API. Remote portlet services don't have to be implemented in Java; they can be implemented in other languages as long as they adhere to the same WSDL description.

Web service providers who want to publish RPWS must publish appropriate entries to a UDDI directory, referencing the RPWS interface WSDL description.

Once a remote portlet has been published, portal administrators can use portal administration tools to search the UDDI directory for Web services that implement the RPWS interface and make some of the matching portlet Web services available to users by adding them to the portal's portlet registry.

Once the portlets are in the registry, users can select them to be displayed on their personal pages.

When a page that references a remote portlet is rendered, the portal uses a portlet proxy to invoke the RPWS. The portlet invokes the portlet proxy exactly as it would invoke a local portlet, passing PortletRequest and PortletResponse objects. The portlet proxy internally invokes a SOAP proxy to marshal all parameters into a SOAP request and sends it to the remote server hosting the RPWS. The SOAP wrapper on the Web service side unmarshals all information in the incoming request and calls on the remote portlet.

For the remote portlet, it's transparent whether it's invoked directly by a portal engine or indirectly through the Web service interface. In each case, it processes the input parameters and returns a PortletResponse object.

The SOAP wrapper marshals the response into a SOAP

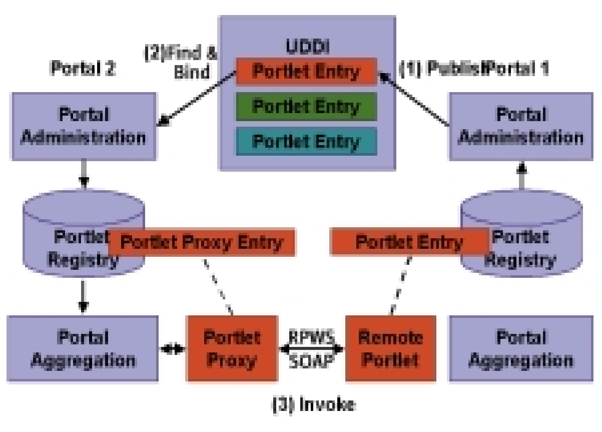


FIG. 8: PROVIDING UDDI CREDENTIALS

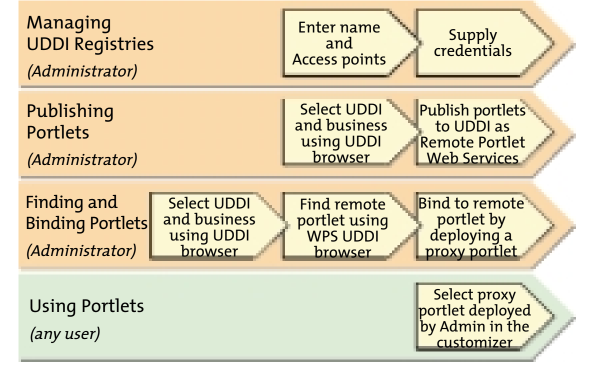

Managing UDDI Registries
(Administrator)
Enter name and Access points
Supply credentials
Publishing Portlets
(Administrator)
Select UDDI and business using UDDI browser
Publish portlets to UDDI as Remote Portlet Web Services
Finding and Binding Portlets
(Administrator)
Select UDDI and business using UDDI browser
Find remote portlet using WPS UDDI browser
Bind to remote portlet by deploying a proxy portlet
Using Portlets
(any user)
Select proxy portlet deployed by Admin in the customizer

FIG. 9: SELECTING THE PORTLET TO BE PUBLISHED

Advertiser Index...

COMPANY	URL	PHONE	PG
CANDLE	WWW.CANDLE.COM/WWW1/WEBSPHERETRIAL	310-535-3600	9
ENGINDATA	WWW.ENGINDATA.COM	201-802-3082	33
GENERAL MAGIC	WWW.GENERALMAGIC.COM	800-468-4342	13
MONGOOSE TECHNOLOGY	WWW.PORTALSTUDIO.COM	281-461-0099	52
PERFORMANT	WWW.PERFORMANT.COM	866-773-6268 EXT. 2	37
PRECISE	WWW.PRECISE.COM/WSDJ	800-310-4777	51
PROLIFICS	WWW.PROLIFICS.COM	800-458-3313	2, 27
SIMPLEX KNOWLEDGE COMPANY	WWW.SKC.COM/TRAINING/PW420	845-620-3700	45
SITRAKA JCLASS SERVERVIEWS	WWW.SITRAKA.COM/JCLASS/WDJ	800-663-4723	4
SITRAKA	WWW.SITRAKA.COM/PERFORMANCE/WDJ	800-663-4723	15
SYS-CON EVENTS	WWW.SYS-CON.COM	201-802-3069	23
SYS-CON INDUSTRY NEWSLETTERS	WWW.SYS-CON.COM	201-802-3020	43
SYS-CON PUBLICATIONS	WWW.SYS-CON.COM	888-303-5282	43
WEB SERVICES WORLD TOUR	WWW.SYS-CON.COM	201-802-3069	19
WILY TECHNOLOGY	WWW.WILYTECH.COM	888-GET-WILY	3

Coming Next Month...

INTO THE BLUE...
BY JOE FARSETTA
What you need to know when you install WebSphere Portal

FINDING THE POT OF GOLD: MAXIMIZING J2EE PERFORMANCE...
BY CODY MENARD
Produce immediate, significant, measurable improvements

DEVELOPING WEB SERVICES WITH WEBSPHERE STUDIO - PART 2...
BY RON BEN-NATAN
Discovering and invoking the Web service

AUTHENTICATION IN WEBSPHERE PORTAL... BY INGO SCHUSTER
The full SSO experience

response and sends it back as the reply to the SOAP proxy, which in turn unmarshals the response for the portlet proxy that finally returns a PortletResponse object to the portal engine that initiated the request.

Web Services in IBM WebSphere Portal

The mechanisms for publishing portlets as RPWS, finding RPWS, binding to them, and using remote portlets must be integrated seamlessly into portal products. Five different dialog flows need to be provided (see Figure 6).

- **_Registering a UDDI server:_** Administrators manage a list of UDDI servers they wish to use for querying and publishing. For each server, WebSphere Portal manages the inquiry/publish URLs, user credentials, and tModel key of the RPWS interface.
- **_Publishing portlets:_** Administrators publish portlets to make them available for use by other portals as RPWS.
- **_Finding and binding portlets:_** Administrators find RPWS and bind to them.
- **_Using remote portlets:_** Users select and use remote portlets transparently, just as easily as local portlets.

Before publishing portlets or looking for remote portlets, one or more UDDI servers must be configured. WebSphere Portal manages these configurations, stores the user name and password securely in the system-wide credential vault, and makes use of the registered servers in the publish-and-find dialogs.

REGISTERING A UDDI SERVER

The administrator can register a new UDDI server using the "Manage Web services" dialog. WebSphere Portal prompts for a display name for this registry, the inquiry URL and the RPWS tModel key that is valid in the scope of the UDDI server. If the server is used for publishing as well, a publish URL can be supplied.

Portlets are published via the "Publish portlets" dialog. The user can select one or more portlets to be published using the standard "Get portlets" dialog. By default, the name under which the portlet is published to the UDDI registry is taken from its registration information within WebSphere Portal. However, the administrator can modify the name and description for each portlet per supported locale. Once the naming is ready, the administrator selects a UDDI registry to publish to. This registry must have already been configured via the "Manage Web services" dialog. The publishing dialog automatically displays a list of all business entities in the selected UDDI registry that the current user has the right to publish to. If none of the existing businesses are appropriate or if there is no such business entity, the user can create a new business on the fly.

Finding and integrating remote Web services as portlets is simple: the "Integrate a Web service as a remote portlet"option lets the user select one of the preconfigured UDDI registries to search in. The administrator can search for all portlets, all portlets in a business entity, or portlets by name. The dialog will find only those portlets that are compliant with the RPWS interface by making use of the associated tModel key.

From the search results, the administrator can select the RPWS to integrate. Finally, WebSphere Portal adds the remote portlet to its portlet registry, making it available for portal users.

Using a remote portlet is as simple as using a locally installed portlet – users can select remote portlets in the "Work with pages" dialog.

PUBLISHING PORTLETS AS REMOTE PORTLET WEB SERVICES IN UDDI

Only portal administrators are allowed to publish portlets as RPWS into a UDDI directory. After logging in for the first time, the administrator must configure a list of UDDI registries to use by selecting the "Manage Web services" dialog (see Figure 7). For security reasons, user credentials are omitted in this dialog. They're only required for the publish operation and can be ignored for the integration and inquiry operations. The "Provide registry authentication information" dialog lets the administrator provide the required credentials (see Figure 8).

To make a portlet available in UDDI the administrator uses the "Publish portlets" dialog. He or she can select the portlet or portlets to publish in addition to the destination registry and business (see Figure 9).

To publish a particular portlet to UDDI as an RPWS, the administrator presses the Publish button.

FINDING AND BINDING TO REMOTE PORTLET WEB SERVICES

Finding and binding to RPWS is allowed for administrators. To find an RPWS, the administrator selects the "Integrate a Web service as remote portlet" dialog.

When the administrator presses the "OK" button, WebSphere Portal queries the UDDI directory for all RPWS that meet the search criteria.

To add the remote portlet to the WebSphere Portal portlet registry and make it available to users, the administrator checks the portlets from the search result list and presses the "OK" button. As a result, WebSphere Portal gets the relevant information about the RPWS and creates a new portlet proxy entry in its portlet registry to make the remote portlet available to users.

In addition, the administrator can specify whether or not user information may be transferred to the RPWS.

USING REMOTE PORTLET WEB SERVICES

For the user, an RPWS is entirely transparent. After logging in, the user can click on the "Work with Pages" link to navigate to the WebSphere Portal Customizer screen.

This screen lets the user select portlets and put them onto a selected page. The screenshot demonstrates that an integrated remote portlet (IBM Proxy Portlet World Clock) is handled exactly like a local portlet (Welcome or Dilbert portlet).

The user can select a proxy for a remote portlet like any local portlet. After the remote portlet is selected in the customizer, it's displayed on the user's page. The left portlet in this screen is remote; the right one comes from a local source.

Conclusion

The concept of remote portlet Web services allows deployment of distributed portals cooperating within an intranet or over the Internet. IBM WebSphere Portal 4.1 supports this concept to allow publishing of local portlets as RPWS and integration of RPWS in WebSphere Portal–based portals with just a few clicks by an administrator, without programming.

At the same time, the ability to host portlets and publish them as RPWS that can be integrated into portals will easily turn WebSphere Portal into a platform that allows content and application providers to offer their services to their customers – portals – in the most easily consumable form.

e-BUSINESS SOLUTIONS: TAKING BUSINESS TO THE NET
EXCLUSIVE INTERVIEWS!
WebSphere DEVELOPER'S JOURNAL
INTRODUCTORY CHARTER SUBSCRIPTION SAVE $30 Today!
Iterative Development with VisualAge for Java
The World's Leading Independent WebSphere Developer Resource

A Conversation with John Shedletsky

*Recently Jack Martin, editor-in-chief of **WSDJ**, spoke to John Shedletsky, vice president of Competitive Technologies at IBM. In addition to running the Competitive Technology Lab, Shedletsky and his team host competitive seminars for business partners and potential customers to illustrate the technical differences between WebSphere and its competition.*

WSDJ: WHEN YOU SAY YOU TRY TO UNDERSTAND COMPETITORS' TECHNOLOGY AND IBM'S TECHNOLOGY, WHAT DOES THAT MEAN?

JS: For example, with BEA WebLogic Server, we got the code, set it up, tried to write applications with BEA tools, and ran the applications on the WebLogic Server. As we do this, we take note of how easy or hard it is, what features are missing, where we ran into trouble. We do the same thing on WebSphere. Part of the exercise is comparisons. It's amazing what you learn.

WSDJ: FOR EXAMPLE?

JS: We found that we can scale better than WebLogic Server can. When it comes to clustered, multitier configurations. WebSphere's features become more pronounced vis a vis WebLogic Server. We're a technology company, and people here have deep technical skills. We're comfortable with and understand the issues that come up in enterprise-scale configurations.

Also, in the WebLogic Server documentation, they talk about uptime and continuous operation and how you can change configuration parameters without having to bring the system down. Well, we actually tried it and it turned out that almost no parameters could be changed without having to restart the machine.

We call it the triangle demo because in their admin console there are these little triangles by most parameters. What does that mean? You try to change the parameter and the triangle starts flashing. It says that you have to reboot the machine for this value to take effect. If you have a cluster of lots of machines, you can imagine this gets to be kind of complicated. That's a demo we usually do just to show people that there is a difference between what is said in the literature and what the actual product can do.

WSDJ: HOW DOES .NET FIT INTO THE PICTURE?

JS: .NET hasn't stepped up to transaction processing yet, they're still using the previous wave technology. Implementations come in waves, and Microsoft has gone through a lot of waves. They had the desktop wave, then there was OLE, a technology to allow these applications to access things on servers. Then they moved into something called COM, then into COM+. This is where they implemented server-based transaction processing. That evolved into something called DNA, Digital Nervous System. Each was a wave of technical implementation. COM+ was the previous wave of how they do transaction processing.

Today, even though they've come out with .NET, it doesn't deal specifically with transaction processing, clustered scalability, and efficient data access. While you can cobble something together, they haven't systematically embedded the capabilities into this .NET architecture. To do transaction processing they more or less rely on what already exists in COM+.

WSDJ: HOW DOES IBM ADDRESS THAT?

JS: We follow these waves. Right now the wave is with EJBs and J2EE. We help with standards in terms of contributing our expertise, and we've been timely in supporting these standards as they come out.

Look into some of the implementation details, where the caching is, the way we do clustering and support easy scalability while supporting a declarative transaction model that makes programming easier and more productive. It's what people at IBM know, and when they build a new transaction processing implementation, they just do it.

WSDJ: IBM KNOWS WHAT THE CUSTOMER WANTS.

JS: Yes. I think that if you look at all the transactions executed on a given day, the vast majority are executed on IBM systems. If you look at all the business data that exists, most of it is still on IBM systems.

We know how to protect that data, keep it valid, do transactions quickly and efficiently, and how to recover when something crashes. We've been doing this for a long time. The people who have worked on those systems make them work in real-world situations where things aren't perfect, where things are down and transactions abort and so forth. These are the same people that have built our support for EJBs.

I would say the same thing about JMS. In some sense IBM invented this idea of messaging and it came out of requirements, in the CICS world. We had these CICS systems and a customer said, "I need a way to asynchronously tie these CICS systems together." It was the Hursley Lab that invented MQ Series to do this. It was totally driven by customer requirement, not people in the lab.

They implemented MQ Series and then it found much wider applicability; now that model has moved into the Java world. The interface has become JMS.

Much of what we learned in implementing MQ Series found its way into the JMS specs because people realized it's a good, robust model. The code used to implement JMS was written by the same people who wrote MQ Series.

It's an example of how the technical concepts don't change that often, although implementations change frequently. Sometimes new things happen, like the Internet happened and everything adapts around that. Now we have protocols that flow across the Internet, which we didn't have before.

Incidentally, we're betting that Web services is the next important protocol. Some say, "Why are you helping Microsoft perfect this Web services protocol?" We see that we're going to get some benefit out of it, and I think they probably see it the same way. It's a win, win.

WSDJ: THERE'S A NEW LEVEL OF COOPERATION BETWEEN IBM AND MICROSOFT?

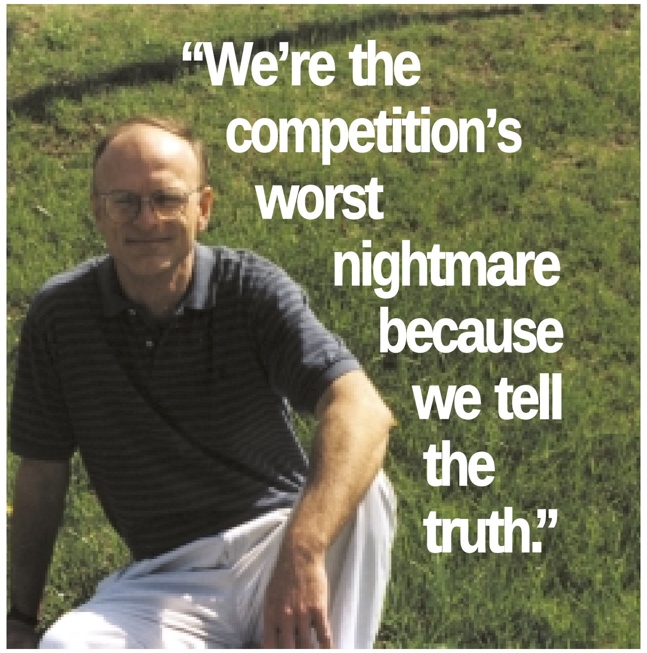
JS: Both companies are realizing that there are areas in which we can cooperate, primarily on the standards. Web services and Web services protocols are good examples. They realize that they have to be able to deal better with the idea that they can inter-operate with other systems in the enterprise and on the Internet. So, a standard protocol will do that. The negative for them, one negative, which turns out to be a good thing for us, is that they have to open up. Before you had a COM object, and DCOM was the way that you could access it remotely and it had a lot of problems. Now they have .NET objects. It's a new object model and the way you remotely access that object is with SOAP. So, they have taken the Web service protocol standard and built it into the new object. That means when you are building a client application, which Microsoft loves to do, it's going to use Web services to talk to the server component. For the first time in Microsoft's history their connection between clients and servers is based on an open standard.

I think it's great for the J2EE world, because supporting Web services means that we have now opened up the server tier. It's no longer chained to the Microsoft clients. You can have more choice in what you implement in the server tier. We follow the standard Web services protocol to talk to clients, whether they're locally attached or across the Internet. Now, customers can decide, "What's the best server infrastructure?" It doesn't have to be based on what talks to the Microsoft client.

WSDJ: SO IT'S NOW BASED ON MY REQUIREMENTS, RATHER THAN MICROSOFT'S REQUIREMENTS.

JS: Based on your requirements. What kind of server platforms do I have? Will I want to do consolidation in the future? Do I want proven scalability? Do I want something that's easy to manage? Do I want to reduce costs by doing consolidation, by running on larger servers rather than lots of little servers? Do I want to outsource it and put up an outsource-hosting environment for Web services that will be provided to my clients? All these are now possible. They happen to work against Microsoft's server software because it becomes just one of many.

Now there are two server software camps. The one camp, J2EE, has 22 licensed J2EE vendors, with WebSphere and WebLogic being the largest, but there are other vendors to choose from



"We're the competition's worst nightmare because we tell the truth."

that are following the J2EE program model. Then you have the other camp, with only one member. .NET basically has a generation-old transaction technology, a rigid view of how to put together multi-tier clusters, and a murky at best story about how to manage them and scale them up. Definitely not able to give you a choice of what kind of servers you want to run on, and if you ever want to do server consolidation or Linux or hosting services, it limits your choice because it's Windows only.

WSDJ: LET'S SHIFT GEARS A LITTLE AND TALK MORE SPECIFICALLY ABOUT PORTAL. WHAT'S WEBSPHERE'S STRONGEST CARD?

JS: B2E, and to accomplish that you have to do things like employee collaboration, employee access to data, Sametime instant messaging, and all the things that Lotus software does – team rooms, and so forth. We support that. We also support Exchange, so through your portal, you can look at your calendar and do your mail with a browser, whether it be Domino or Exchange. In addition, we give users access to back-end systems like Siebel and SAP.

Through a portlet you can access these back-end systems. The WebSphere Portal supports mobile and pervasive devices easily. We use style sheet translations or you can generate markup language for the specific device. Then the portal can be enhanced with things like the voice server so that you can hear the spoken word and speak back. You can use the same portal infrastructure for a voice channel to the customer, as well as a browser channel. Our portal supports machine translation as well, if you want to use that feature.

WEBSPHERE COMPETITIVE ADVANTAGES

- Better performance
- Lower license and support cost
- Be twice as productive with WebSphere tools
- Richer portals, including B2E
- Wireless support built-in
- Closed-loop merchandising
- Voice support and machine translation built-in
- Message backbone capability
- More productive J2EE/CA tools
- More comprehensive XML and Web services tools

When you stack up all the features in our portal, it's just far beyond the capabilities of the other platform vendors. By that I mean BEA, Oracle, and Microsoft.

WSDJ: HOW DO YOU SEE THE SAP COMPETITIVE OFFER?

JS: Because SAP is in a lot of businesses, it's natural to talk to SAP; they say, "I can give you a portal and extend it." The problem with SAP is that they write everything themselves. Why in the world would SAP think they could write an application server better than IBM, BEA, or Microsoft? But they are. They are writing the connection software. They are writing everything themselves.

You have to wonder how they're going to keep up. To some extent their main advantage was giving you access to the SAP ERP systems. But you can get that with WebSphere Portal.

I think it's amazing how this concept of a portal is galvanizing different teams in IBM and they're all coming together with technology for the WebSphere Portal Server. I think it's going to give customers a tremendous range of options.

WSDJ: CAN YOU ENCAPSULATE THE PORTAL CONCEPT FROM THE TECHNOLOGY AND MARKETING SIDES AND GIVE US YOUR SYNOPSIS?

JS: Imagine a business and an Internet channel where you ask visitors, "How can I improve your experience? How can I be more useful?" They might say, "Show me what I want to see. Show me only what I'm interested in. Accommodate my lifestyle because I'm not in the office all the time. Speak my language." That's what the portal does. Because of the personalization in the portal you can do one-to-one merchandising. They will see stuff that's of interest to them. To accommodate their lifestyle, it means that maybe you have to support cell phone access to the portal. Maybe you have to have telephone support. Then obviously, you've got to speak their language. So you need globalization or even on-the-spot machine translation.

WSDJ: HOW DO WEBSPHERE TOOLS COMPARE TO BEA'S?

JS: BEA doesn't really have a toolset. They have some piece part tools. WebLogic Workshop is the latest example of a Web services tool. They have relied on WebGain, which in turn relies on Dreamweaver and Visual Café. However, it looks like WebGain is going out of business. I think I'd be worried if I were BEA. They have no tools strategy.

WSDJ: I'M CURIOUS ABOUT HOW YOU SPEND A TYPICAL DAY. WHAT CAN YOU TELL ME ABOUT HOW YOU AND YOUR GROUP SPEND YOUR TIME?

JS: We do bench work, both in terms of feature function and sometimes performance comparisons. Across the team we have work going on in .NET, WebLogic, and business integration vendors. Right now the focus is on Tibco.

We look at the base stuff, what we call the foundation, so it's base application servers. We look at reach and user experience and then we look at business integration. The interesting thing is that you have to spend half your time on the IBM products in order to understand the comparisons when you're looking at the competitor's products.

We're concerned about taking the full IBM platform, putting it together, and showing people how it works to solve a customer's problem. That's really about half of what we do.

The rest of my time I spend figuring out how to communicate it. We write papers, we do Web conferences, but the most effective technique that we've found is inviting people to events, classes, and seminars where we go through the whole platform.

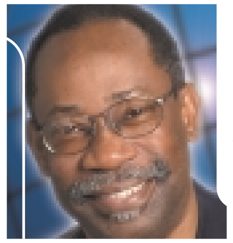
We're the competition's worst nightmare because we tell the truth. I believe Harry Truman said, "I don't give 'em hell. I just tell the truth and they think it's hell."

For more information, see www-3.ibm.com/software/info1/websphere/partners/index.jsp?tab=ed/competitiveroadshow

Finding the Pot of Gold

Maximizing J2EE Performance

BY CODY MENARD



There are some undeniable truths about the state of IT operations today and their direction for the future. The recent recessionary environment, coupled with IT talent shortages that already existed, has turned the IT operations group into an embattled organization – greater and greater responsibilities with fewer and less-trained resources to manage very complex, n-tier business systems. IT administrators need all the help they can get from the Enterprise Systems Management vendor community; not just more monitoring software ("Yes I can see it, but what does it all mean?"), but rather an intelligent and active management approach.

Applications that are crucial to business success have been moved online. This online trend has been driven by business objectives like generating and protecting revenue or improving margins by moving customers to more efficient Internet-based channels. Especially for mission-critical online business systems, there is the confounding circumstance of highly complex, multitiered IT environments that, as a result of explosive surges in transaction volumes and continual technological advances, have become monumentally more difficult to manage. A typical system could include firewalls, routers, HTTP load balancers, and gigabit ethernet switches at the network layer, with Web server clusters, multiple application server clusters, message-oriented middleware (MOM) servers, legacy gateways, and mid-tier database clusters for the application infrastructure. To make matters even more complex, the added complication of component programming is that the way you deploy components (deployment descriptors) affects performance and availability of your application as much as the quality of the code itself.

Improving application integration has also become an essential component of contemporary IT strategy. In a recent Morgan Stanley survey, 225 CIOs – a clear 80% – indicated that they would begin new application projects in 2002 with application integration as their top priority (eclipsing other hot topics such as security and network capacity). Application integration has become a top priority as companies experience escalating economic pressure to mine additional productivity from existing, disparate but related applications, legacy systems, and new Web services, while ensuring that they all function faultlessly together.

Where there is challenge, there is opportunity. The challenge is substantial – complexity is making it more difficult to manage any business system as a cohesive computing entity, creating the possibility for revenue loss through dropped transactions and expense overruns as customers revert to more expensive offline channels and IT personnel take exponentially more time to fix infrastructure problems. Unfortunately, status quo for most companies, by either design or legacy, is to have separate monitoring tools for each component. This requires that a human being review the output across the different servers and sub-components to determine what is merely a symptom and what is the actual cause of the problem. Customers and analysts agree: many of the current management solutions remain part of the problem, since they actually increase time-to-resolution by deluging users with hundreds or thousands of datapoints – rather than answers.

The opportunity: find a tool that reduces the time and difficulty experienced when managing a complex n-tier environment. Analyst firm Enterprise Management Associates believes that real-time management solutions need to meet three requirements to be effective. First, the solution must apply across all the interrelated components in real-time via some form of automated correlation. Second, it must bring immediate value with embedded expertise on how to fix the performance or availability problem. Third, it must account for the reality that all implementations include customized applications and IT configurations.

Clearly, that would require a management solution that can automatically detect problem situations across the multiple tiers and subcomponents (domains) of the business system infrastructure, quickly isolating the fault and suggesting exact fixes.

Even better, what if the management software could actually invoke those fixes or preventative actions across the different components once problems are detected? An intelligent and active management approach like this would ensure that online revenue continues flowing and would considerably reduce the difficulty in managing these infrastructures and allow companies to leverage their stretched IT resources more effectively. This is truly the blueprint for the "next-generation" applications management solution so desperately needed right now.

The true next-generation applications management solution provides immediate cross-domain fault and performance isolation and resolution. Problems as varied as excessive network latency between clustered application servers or purchase orders held up in an asynchronous message queue can be automatically fixed or prevented when detected, resulting in greater online transaction throughput, direct expense reductions, and drastically reduced downtime.

Implementing a next-generation J2EE management solution would produce immediate, significant, measurable improvements. It's the only real way to maximize your J2EE performance and substantially accelerate the ROI on your investment. It would be like finding a pot of gold for your company.

ABOUT THE AUTHOR... Cody Menard is a founder and CTO of Covasoft, Inc. (www.covasoft.com), and has extensive experience in software integration, development, and management. His expertise in systems management, neural networks, artificial intelligence, and J2EE led him to create a novel approach to the challenge of managing complex systems and applications based on relationships across infrastructure domains.

E-MAIL
cmenard@covasoft.com