INTERVIEW: CHIEF ARCHITECT OF WEBSPHERE PORTAL 4.1 PAGE 16

WebSphere Developer's Journal

The World's Leading Independent WebSphere Developer Resource

WebSphereDevelopersJournal.com

JUNE VOLUME 1 ISSUE 5

ANNOUNCING WEB SERVICES EDGE WEST COAST! OCT. 1–3, 2002 SAN JOSE, CA

web services EDGE conference & expo

SAN JOSE MCENERY CONVETION CENTER

SEE PAGE 23 FOR DETAILS

SPECIAL PORTAL FOCUS ISSUE

SYS-CON MEDIA

PROLIFICS

WWW.PROLIFICS.COM

WILY TECHNOLOGY

WWW.WILYTECH.COM

SITRAKA JCLASS SERVERVIEWS

WWW.SITRAKA.COM/JCLASS/WDJ

WebSphere® DEVELOPER'S JOURNAL

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
Peter Fischer, David Gootzit, Stefan Hepper, Stephan Hesmer, Carol Jones, Jack Martin, Chris Paul, Gene Phifer, Thomas Schaeck, Sung-Ik Son, Robert Will

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: $12 U.S. $15 All others

PUBLISHER, PRESIDENT AND CEO FUAT A. KIRCAALI
CHIEF OPERATING OFFICER MARK HARABEDIAN
CHIEF FINANCIAL OFFICER BRUCE KANNER
VP, BUSINESS DEVELOPMENT GRISHA DAVIDA
SENIOR VP, SALES & MARKETING CARMEN GONZALEZ
VP, PRODUCTION 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, 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
JDJSTORE.COM ANTHONY D. SPITZER
CUSTOMER SERVICE MARGIE DOWNS

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

It's About Time and Money and User Experience

BY JACK MARTIN

Putting this issue together was a great experience for me. I've met a fine bunch of people who make up the WebSphere Portal team. I've heard time and again how, working together, the team made tremendous strides in creating a superior software suite in record-breaking time.

On top of the Portal heap is Larry Bowden, vice president of WebSphere Portal. His previous experience ran across numerous IBM divisions, so although I joked with him about leveraging his career portfolio the same way he is leveraging the WebSphere Portal brand, in my opinion it was key. Bowden's broad experience across different business functions parallels the nature of Portal.

He had to interface with a lot of different groups inside IBM. His team collected code from various sources within Tivoli, Lotus, WebSphere, and DB2, and from many different laboratories around the world. IBM then built brand-new code into the Portal product based on extensive market research and analysis. This is a fast approach to delivering software to market.

IBM didn't develop and build an internal view of what the market wants and then see if someone would buy and use it. What they did instead was go in the opposite direction, believing that the number one thing is to see through the customers' eyes the product that they want. In an environment where not everyone was sure what a portal was yet, what sense did it make to create something – leaving a hit or miss to chance? None. The simplest things are sometimes the most profound.

I see tremendous opportunities with WebSphere Portal. It's a solid, scalable, robust portal platform. Lots of ISVs already see that they can embed it in their solutions. Of course, IBM knows that what's good for the developer community is good for achieving a bigger chunk of market share.

ISVs don't want and don't have to spend their resources re-creating the middleware. They can invest in their domain expertise. And IBM, which isn't in the application business, puts all of its resources into the middleware to create the most solid foundation there is. ISVs are applying all of their forces to applications, and IBM is applying all of its forces to middleware. It's possibly the best win-win situation I've ever seen.

Then there's the impact WebSphere Portal has on the end user. A ubiquitous user experience at the desktop means that you can't be selective as to what you want to access through Portal. Now, the power of that interface is being turned over to the users. If users want to see five things at a time in one panel, they can have that. Then they can go to another page with three different things, and if that's how they'd like to work, they can just set it up that way. And while users need access to certain data, no one has predetermined how it will show up on their desktop.

Users in shipping and receiving are going to have a different interface – one constructed for their way of interacting with it. It'll be different from that of the users in HR and different from that of the users on the manufacturing floor. The experience from the portal is so tailored to the individual that they'll think they're working in a totally different product. It's a single point of personalized access where the applications, the content, the people, and the processes are managed and surfaced within that portal.

The sales force can construct it uniquely to see their customers' stock prices, what their sales were last week, what their budgets look like, and when their meetings are – it becomes their personalized environment. The sales guy is saying, "This is exactly what I wanted. IT would never build me anything that customized."

The IT organization in a company wants to cut costs, they want to get a common infrastructure, and they want centralized control, security, and audits. Portal gives them a horizontal platform allowing them to do that. When IT looks at Portal, they see a maintainable environment, a centralized infrastructure, and a single sign-on.

Most impressive is the vision that over three years, by going to a Portal environment from the current one, there is a total cost of ownership reduction of an astonishing 60%. Amazing.

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

Portlet development and testing

WebSphere Portal Mobile Access Portlets for WAP Devices

BY SUNG-IK **SON**

IBM's WebSphere Portal currently supports mobile devices by generating portal pages in three markup languages: HTML for desktop computers and some PDAs, WML for WAP devices, and cHTML for devices in the NTT DoCoMo i-mode network. This article focuses on portlet development for WAP devices, including a brief introduction to WAP and WML, and discusses various WAP emulators for portlet testing. The article's source code is not necessarily driven by best practices or performance considerations.

ABOUT THE AUTHOR

Sung-Ik Son is a senior software engineer in the application and integration middleware division of IBM's Raleigh Lab. For more than 12 years, he's been a key developer in IBM's system and application software development organizations; for the past three years he's worked on the WebSphere Enablement and Consulting team. Sung-Ik is a coauthor of the IBM Redbook *WebSphere V3.5 Handbook.*

E-MAIL

sungik@us.ibm.com

WAP and WML

WAP (Wireless Application Protocol) is a communication protocol designed for wireless devices just as HTTP is for desktop Web browsers. WML (Wireless Markup Language) is a markup language for wireless device microbrowsers as HTML is for browsers on desktop computers. WML content is read and interpreted by a microbrowser built into the WAP device. WML is similar to HTML, but less forgiving because it was designed using XML as a base.

Decks are WML documents and each deck consists of one or more cards. Decks begin with the <wml> and end with the </wml> tag, and cards begin with the <card> and end with the </card> tag. The WAP device's microbrowser reads the whole deck while accessing it in order to navigate its cards without having to load more data. Once a deck is loaded, the cards inside the deck remain in the WML microbrowser's memory until the browser is asked to reload the whole deck. Listing 1 (Listings for this article can be found at www.sys con.com/ websphere /sourcec.cfm) will give you a taste of WML and WAP devices; Figure 1 shows the output on a RIM BlackBerry device.

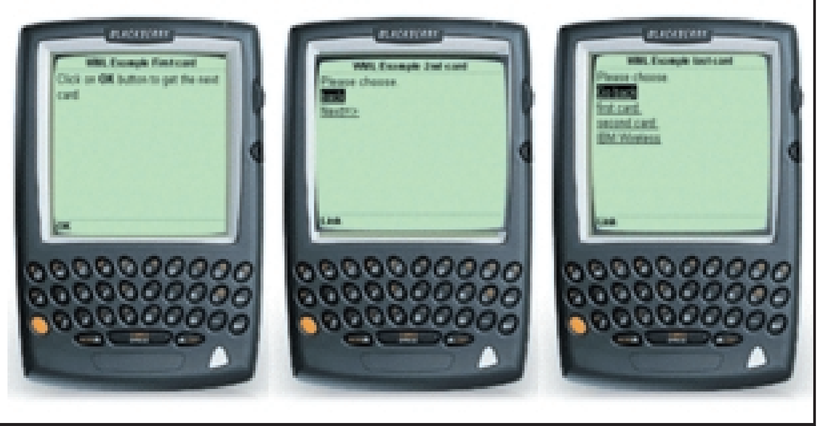


FIG. 1: OUTPUT DISPLAYED ON THE RIM BLACKBERRY DEVICE

Mobile Access Portlets for WAP Devices

Now that you have an understanding of WAP and WML, we're ready to develop the following portlets:

- Personal bookmark–setting
- WAP URL surfing
- XML and XSLT
- User-agent displaying

PERSONAL BOOKMARK-SETTING PORTLET

The personal bookmark–setting portlet can configure an unlimited number of bookmarks to be inserted and deleted, for example, to the portlet deployment descriptor (portlet.xml) file. To develop the portlet, you need to understand the WML <anchor> tag. A WML card can be set up to display WML's anchor functions. The <anchor> tag always has a specified task ("go," "prev," or "refresh") that defines what should be done when the user selects the link. In the example shown in Listing 2, when the user selects the "Die Bahn" link, the task goes to http://wap.bahn.de.

Note that I'm using the Bookmark-Portlet_04.java example described in the Portlet Development Guide, Introduction to the Portlet API (see the references section). The portlet deployment descriptor is shown in Listing 3 – I've slightly modified the program for WAP device support application; the modified portion appears in bold type. Listing 4 shows the source portlet WAPBookmarkPortlet_04.java. The output is shown in Figure 2.

WAP SITE SURFING ON THE INTERNET

The WAP URL surfing portlet provides unrestricted WAP site surfing. The user enters any URL of a WAP site. Once the portlet accesses the first page of an external WAP site, consequent links will work as long as the subsequent WML pages are well formed. You can arrange the predefined site with the bookmark-setting portlet. In addition to the following code, see Listings 5 and 6.

```
<jsp:useBean id="url"
  class="java.lang.String"
  scope="request" />
<p>
<anchor><go
  href="http://<%=url%>"/>
  http://<%=url%></anchor>
  <br/>
</p>
```

The output is shown in Figure 3.

XML ACCESS URL PORTLET

The XML access URL portlet can access an external site that produces XML content. Using stylesheets, the same XML content can be displayed in a variety of formats applicable for different browsers and devices. This way, a single content source can be used to create content in different formats, such as WML, cHTML, and HTML (see Listings 7 and 8). In this case, you don't have to keep multiple content sources based on different markup languages. In this example the XML content will be applied by an XSLT stylesheet and converted to WML. Note that although I use the FlightInfo.xml and FlightInfoForWML.xsl files (see Listings 9 and 10) as provided by the WebSphere Transcoding Publisher, I developed the other necessary source code using XML and XSL stylesheets. The portlet will use the following XML and XSLT sources:

- XML URL: http://home.nc.rr.com /sungik/flightinfo.xml
- XSLT URL : Inside the portlet

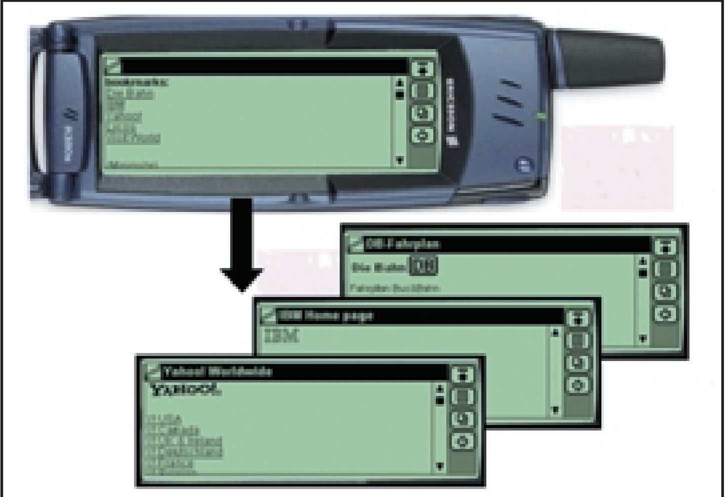


FIG. 2: OUTPUT DISPLAYED BY DIFFERENT DEVICES

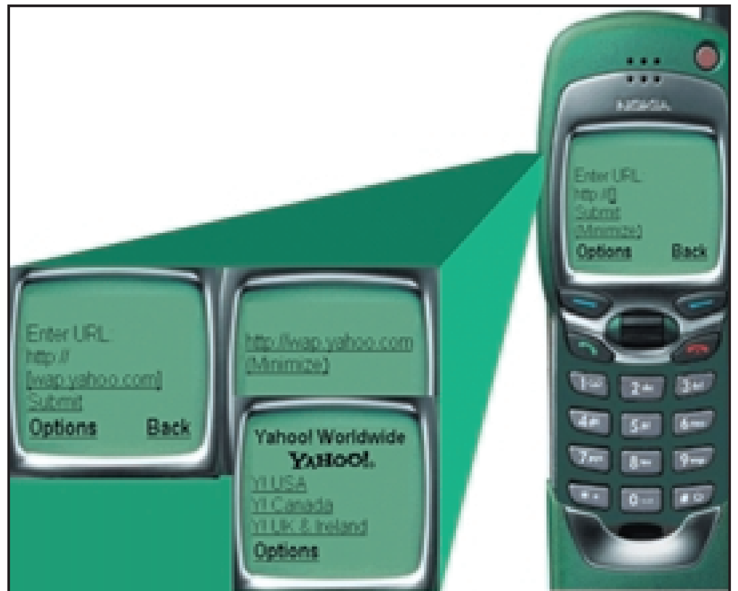
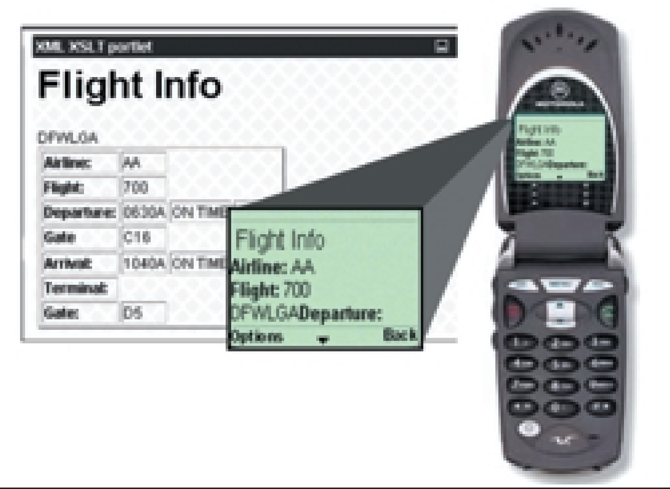


FIG. 3: OUTPUT DISPLAYED BY NOKIA AND OPENWAVE DEVICES



XML XSLT portlet

Flight Info

DFWLGA

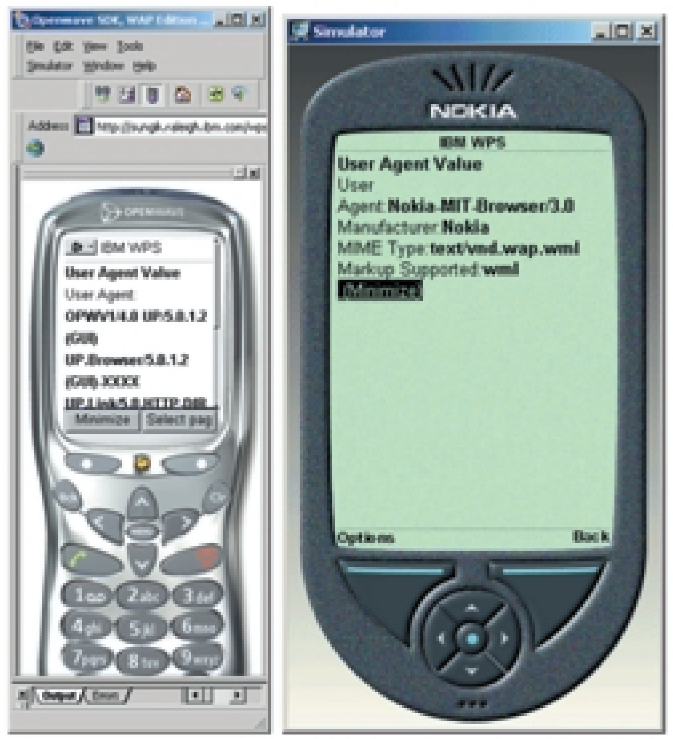
Airline:	AA	
Flight:	700	
Departure:	0630A	ON TIME
Gate:	C16	
Arrival:	1040A	ON TIME
Terminal:		
Gate:	D5	

Flight Info
Airline: AA
Flight: 700
DFWLGADeparture:
Options Back

FIG. 4: MOBILE ACCESS PORTLET TESTING USING WAP DEVICE EMULATORS

FOR MORE INFORMATION

- *"Portlet Development Guide, Introduction to the Portlet API":* www-3.ibm.com/software/web-servers/portal/doc/PortletDevelopmentGuide.doc.
- *Nokia WAP Toolkit:* www.nokia.com/corporate/wap/sdk.html.
- *Palm OS 3.1 Emulator:* www.palmos.com/dev/tools/emulator.
- *Openwave SDK WAP Edition:* (http://updev.phone.com)
- *Klondike WAP Browser:* www.apachesoftware.com/download.html.
- *Blazer Palm Browser:* www.handspring.com/software/blazer_overview.jhtml.
- *The AnywhereYou Go.com WAP browser, powered by 4thPass:* www.anywhereyougo.com/images/palm35.zip.
- *Yahoo Germany:* http://de.mobile.yahoo.com.
- *"WebSphere Solutions for XML, XSL Stylesheet, and Transcoding for Web Application Data Accessible by Various Devices":* www7b.boulder.ibm.com/wsdd/techjournal/0111_son/son. html.


IBM WPS
User Agent Value
User Agent:
OPWV1/4.0 UP/5.0.1.2
(GUI)
UP.Browser/5.0.1.2
(GUI) XXXX
UP.Link/5.0.HTTP-DIR

IBM WPS
User Agent Value
User
Agent: Nokia-MIT-Browser/3.0
Manufacturer: Nokia
MIME Type: text/vnd.wap.wml
Markup Supported: wml

FIG. 5: OUTPUT DISPLAYED BY NOKIA AND OPENWAVE DEVICES

FIG. 6: WAP DEVICE EMULATORS SIMULATE WAP DEVICES

The output displayed by the different devices is shown in Figure 4.

PORTLET DISPLAYING USER-AGENT VALUES

A client device with a specific browser sends the user-agent value to the Web server when requesting a Web page. Since WAP devices vary in capability and features such as image support, deck size, and color support, the Web server can use the user-agent value to decipher the specific browser and device making a request, then adjust the content accordingly. Please refer to www.thewirelessfaq.com/useragents.asp for the user-agent values of different devices. The portlet shown in Listing 11 will display the user-agent value to your specific device. Figure 5 shows the output displayed by Nokia and Openwave devices.

Mobile Access Portlets by PDA

To access the WAP portal site using Palm devices, use the Blazer Palm browser, which supports multiple Internet standard protocols including HTML, cHTML, and WML. The AnywhereYouGo.com WAP browser by 4thPass is also good for viewing WAP-enabled sites from any mobile device running the Palm OS. To access the WAP portal site using PocketPC or Win CE devices, use the Klondike WAP Browser. Please refer to the reference section for information on downloading these browsers.

WAP device emulators are software programs that simulate WAP devices on your desktop computer (see Figure 6). The following work well with the IBM WebSphere Portal portlets developed for WAP devices:

- ***Nokia Tool:*** http://forum.nokia.com/main.html. Select Programs –> Nokia Mobile Internet Toolkit –> Mobile Internet Toolkit.
- ***OpenWave SDK WAP Edition 5.0, 22.5 MEG:*** http://developer.openwave.com.
- ***Klondike WAP Browser:*** www.apachesoftware.com/download_kwpe.html.
- ***Yahoo Germany:*** http://de.mobile.yahoo.com.

Conclusion

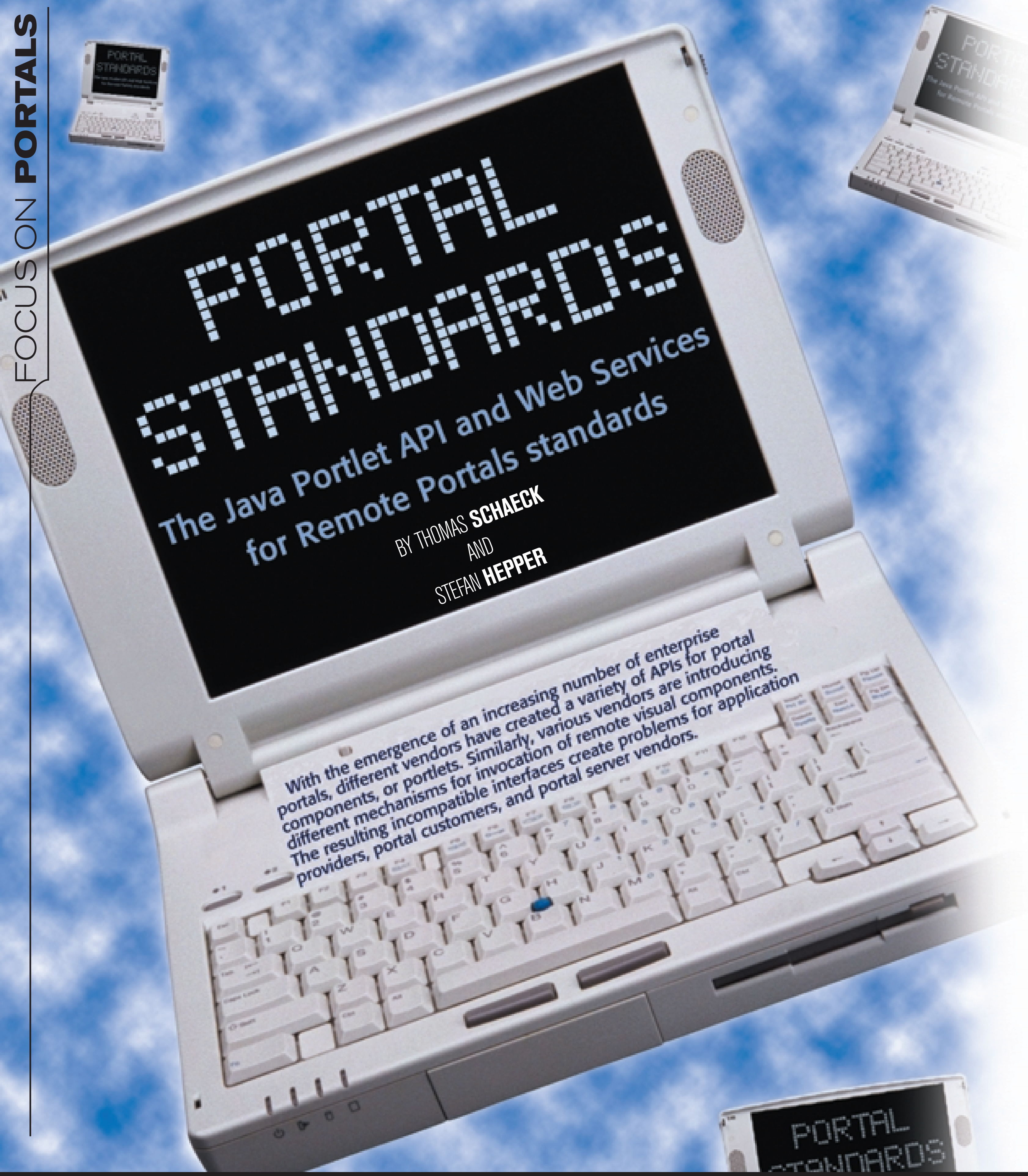
WAP's popularity arises from the fact that it's been adopted and supported by major computing and communications industry players. IBM WebSphere Portal provides developers with the environment to develop WAP applications and build portal Web sites for wireless use. You can develop simple portlets to handle WAP devices and test their outcomes by using various WAP device emulators.

References

IBM products referenced here:
- *WebSphere Portal:* www-3.ibm.com/software/webservers/portal.
- *WebSphere Transcoding Publisher:* www.ibm.com/software/webservers/transcoding.

CANDLE

WWW.CANDLE.COM/WWW1/WEBSPHERETRIAL



PORTAL STANDARDS

The Java Portlet API and Web Services for Remote Portals standards

BY THOMAS SCHAECK AND STEFAN HEPPER

With the emergence of an increasing number of enterprise portals, different vendors have created a variety of APIs for portal components, or portlets. Similarly, various vendors are introducing different mechanisms for invocation of remote visual components. The resulting incompatible interfaces create problems for application providers, portal customers, and portal server vendors.

The Java Portlet API and Web Services for Remote Portals (WSRP) standards will overcome these problems by providing interoperability between portlets and portals as well as between portals and visual, user-facing Web services.

With these standards in place, application providers or portal customers will be able to write portlets or visual, user-facing Web services independent of specific enterprise portal products. The Java Portlet API will be compatible with WSRP and will allow the publishing of portlets as Web services.

IBM initiated both the Java Portlet API and WSRP standards and is committed to implementing these standards in WebSphere Portal. The WebSphere Portal for Multiplatforms 4.1 release provides an API for locally deployed portlets written in Java and a Web services interface for remote portlets that are architecturally consistent with emerging standards.

Portals and Portlets

Portals provide personalized access to information, applications, processes, and people. They're typically based on aggregation of visual components, i.e., portlets. Figure 1 shows an example of an enterprise portal viewed from an HTML browser.

The content displayed in the Web browser can range from the simple (news portlets, search portlets, and stock quote portlets) to the complex, interactive content displayed in Figure 1. These portlets are aggregated on the user's personal page to make relevant information visible in a single view. Users can select different portlets from a pool and put them on their own pages. In our example, the user has created the pages indicated by the tabs on the upper left side: Welcome, Basics, Applications, Collaborate, Service, and Special Portlets.

The market report portlet shown in Figure 2 is of medium complexity. It has a View mode that displays a list of stock quotes; this example is only for the stock of the corporation running the employee portal. If users want to track other stocks, they click on the Edit button in the title bar so the portal displays the Edit mode of the portlet.

To find out how the portlet works, users click the Help button and the portal displays the Help mode of that portlet. The maximize button maximizes the portlet. Each of the portlet's supported modes might consist of one or more fragment views. For example, the View mode in this example consists of fragment views for the stock quotes list and market details.

Although this example only presents the HTML views, the same portlet might also have alternative views producing different markup types to be shown or read when the user uses a WAP phone or PDA or accesses the portal via a normal telephone and a voice gateway.

Portal server implementations typically include a component receiving requests and aggregating personalized content for the user – a component in which portlets run and that provides portlets with the required context. Figure 3 illustrates how portlets are invoked by a J2EE-based portal. In the example shown in Figure 2 the user clicks the market details link in the market report portlet. The browser generates an HTTP request that is received by the portal servlet. The portal servlet invokes code to determine the set of portlets to be displayed for the user, dispatches the request to the targeted action of the market report portlet, and invokes the set of portlets on the page, aggregating their content in a single HTML page that is returned.

Like servlets, portlets are Web components, but they have additional properties allowing them to easily plug in to and run in enclosing Web applications such as portals. Portlets are designed to be aggregated in the larger context of composite pages; multiple instances of the same portlet parameterized with different per-user, per-instance portlet data can coexist on the same portal page. In the course of handling a single request, many portlets are usually invoked to aggregate their produced markup fragments in a portal page. The markup fragments generated by portlets often need links to trigger actions in the portlet; therefore, URL-rewriting methods are required that allow portlets to transparently create links within the markup fragments they output, without needing to know how URLs are structured in a particular Web application.

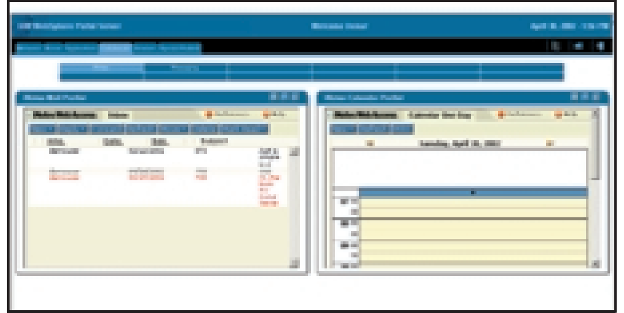
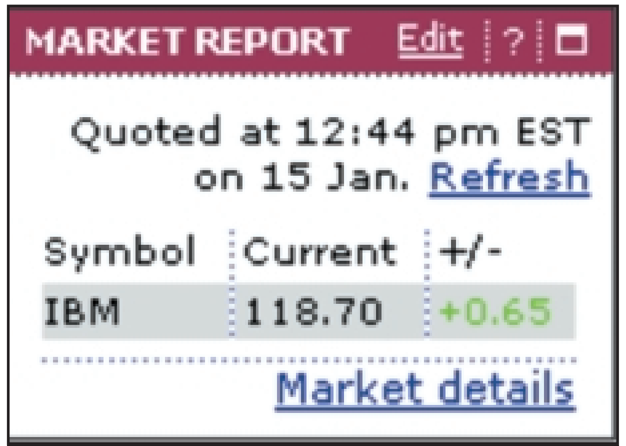


FIG. 1: EXAMPLE OF A PORTAL


MARKET REPORT Edit ?

Quoted at 12:44 pm EST on 15 Jan. Refresh

Symbol	Current	+/-
IBM	118.70	+0.65

Market details

FIG. 2: EXAMPLE OF A PORTLET

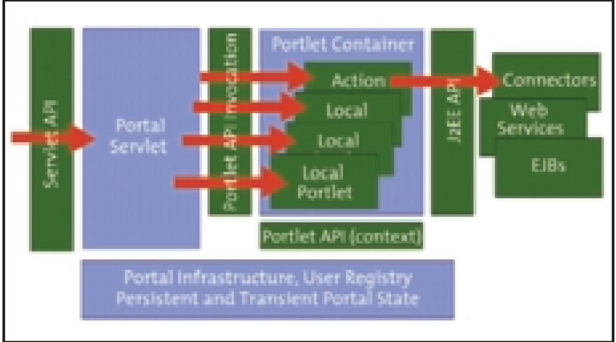


FIG. 3: INVOCATION OF PORTLETS IN A J2EE-BASED PORTAL SERVER

ABOUT THE AUTHORS

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

Stefan Hepper joined IBM in 1998, is now engaged in the IBM WebSphere Portal Server project, and coleads the JSR 168 to standardize the Portlet API. He has lectured at international conferences, published papers, filed patents, and coauthored *Pervasive Computing* (Addison-Wesley 2001).

E-MAIL
schaeck@de.ibm.com
sthepper@de.ibm.com

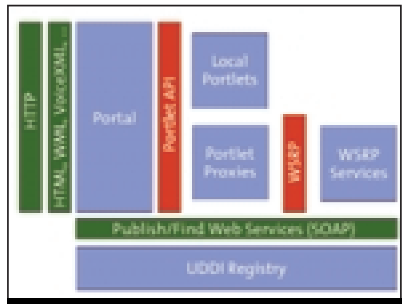


FIG. 4: EXISTING (GREEN) AND MISSING (RED) PORTAL STANDARDS

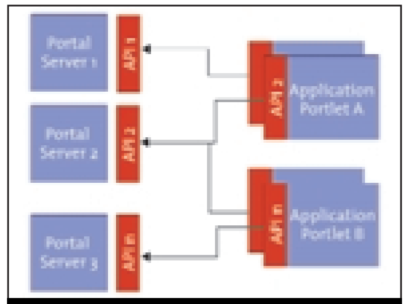


FIG. 5: DIVERSITY OF INTERFACES FOR PORTAL COMPONENTS

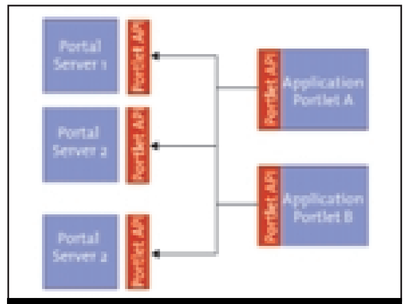


FIG. 6: COMMON PORTLET API FOR PORTAL COMPONENTS

Internally, portlets may be implemented in different ways. For example, they may be implemented according to the Model-View-Controller pattern, where the model and controller are Java classes and the views are JavaServer Pages, XSLT stylesheets, or other kinds of templates.

Portlets rely on container infrastructure accessible via the portlet API for functions like access to user profile information for the current user, access to the window object that represents the window in which the portlet is displayed, participation in the portal window and action event model, access to Web client information, inter-portlet messaging, and a standard way of storing and retrieving persistent per-user/per-instance data.

The Need for Standards

The lack of standards in the portal space has led portal server platform vendors to define proprietary APIs for local portal components and for invocation of remote components. The resulting diversity of interfaces creates interoperability problems for portal customers, application vendors, content providers, and portal software vendors.

Application vendors who want to provide portlets to make their applications accessible through portals need to provide different versions for the different APIs of the different portal server platforms, which results in high development costs.

Customers who develop their own portlets have to invest significant effort into portlet code that will typically run only on the portal server platform on which it was developed. This locks customers into their particular portal server platforms or at least makes migration difficult.

Content providers have no easy means to publish their content in a form that allows it to be found and bound to by portal servers in a plug-and-play fashion.

The smaller portal software vendors have difficulties convincing application providers to provide portlets for their platforms.

Standards exist for both the client-to-portal protocols and the different markups: devices or gateways typically access portal servers via standard HTTP, and depending on the device type, different standardized markup types may be used (HTML for the typical Web browsers, WML for WAP phones, VoiceXML for access from phones via voice and a voice gateway, etc.). Standards also exist – Web Services Description Language (WSDL) to describe Web services; Universal Description, Discovery, and Integration (UDDI) to publish and find Web services in directories; and Simple Object Access Protocol (SOAP) to invoke Web services. Existing standards are illustrated in green in Figure 4.

Missing is a standard for portal components to be deployed locally on portal servers (portlet API) and a standard for remote components that plug and play with portal servers (WSRP). Missing standards are illustrated in red in Figure 4.

Figure 5 depicts the result of the absence of a common API standard. The application providers who provide portlets typically select a few major portal platforms and have to implement more than one version of a particular portlet. From a portal server perspective, only a subset of portlets runs on each individual portal server platform. Portal customers may end up in situations where their required combination of portlets is not available on their selected portal server.

The Java Portlet API solves these problems by establishing interoperability between portlets and portals. As shown in Figure 6, all portlets written to the Java Portlet API will run on all compliant portal servers.

Similarly, the WSRP standard will enable interoperability between portals and WSRP-compliant visual, user-facing Web services for portals.

Portal vendors will be able to provide portal software that can run all compliant portlets, either local portlets written to the Java Portlet API or Web services conforming to the WSRP standard. ISVs, application providers, customers, and content providers will be able to implement portlets or visual, user-facing Web services that can be used by all compliant portals.

Java Portlet API

The Java Portlet API specification defines a Java API for Web application components that interact with and can be aggregated in applications such as portals; we refer to these components as portlets. The Java Portlet API will be defined in the Java Community Process as JSR 168 (www.jcp.org/jsr/detail/168.jsp).

The expert group's initial kick-off meeting took place March 13–15, 2002, in San Francisco. They agreed that a simple but usable portlet API is needed, one similar to the servlet API. Their goal is to define a first public draft by the end of September 2002 and version 1.0 of the specification, the reference implementation, and the compliance test kit by the end of December 2002. It's planned to have the reference implementation done as an Apache Jakarta subproject.

The portlet API will define interfaces for portlets and tags for JSPs called by portlets, cleanly separating portlets from the surrounding portal server infrastructure so they can run on different portal servers the way servlets can run on different application servers.

The portal reference architecture of the JSR is shown in Figure 7. The JSR will only define the portlet API and the

GENERAL MAGIC

WWW.GENERALMAGIC.COM

contracts between the portlet container and the portlet. The portlet container provides the runtime environment for the portlets, just as the servlet container provides the runtime environment for servlets, and will leverage the servlet container wherever possible. The portal Web application sits in front of the portlet/servlet container and is responsible for aggregation, customization, and authorization.

Web Services for Remote Portals (WSRP)

WSRPs are presentation-oriented, user-facing Web services that plug and play with portals or other applications. They are designed to enable businesses to provide content or applications in a form that doesn't require any manual content- or application-specific adaptation by consuming portals; WSRP services can easily be aggregated by portals without programming effort. As WSRP includes presentation, WSRP service providers determine how their content and applications are visualized for end users and to which degree adaptation, transcoding, translation, etc., may be allowed.

WSRP services can be published in public or corporate service directories (UDDI) where they can easily be found by portals that want to display their content. Web application deployment vendors can wrap and adapt their middleware for use in WSRP-compliant services. Vendors of intermediary applications can enable their products for consuming WSRP services.

Using WSRP, portals can easily integrate content and applications from internal and external content providers. The portal administrator simply picks the desired services from a list and integrates them; no programmers are required to tie new content and applications into the portal.

To accomplish these goals, the WSRP standard will define a Web services interface description using WSDL and all the semantics and behavior that Web services and consuming applications must comply with in order to be pluggable, as well as the meta-information that has to be provided when publishing WSRP services in UDDI directories. The standard allows WSRP services to be implemented in very different ways, be it as Java/J2EE-based Web services, Web services implemented on the .NET platform, or portlets published by a portal as WSRP services. The standard enables use of generic adapter code to plug any WSRP service into intermediary applications rather than requiring specific proxy code.

The WSRP specification will permit portals to consume WSRP services by using generic portlet proxies to dynamically plug in any WSRP services without programming effort. It will also permit portals to publish local portlets as WSRP services and will permit implementation of WSRP services on any Web services–capable platform, e.g. J2EE or .NET. It will provide enough information to producing services to allow them to meter and bill if desired.

The initial meeting of the OASIS WSRP technical committee took place at the IBM Reasearch Lab in Hawthorne, New York, March 18–20, 2002. At that meeting, members decided to aim for year's end as the target for providing a 1.0 version of the WSRP specification as well as a compliance test kit. (See http://oasis-open.org/committees/wsrp for more information.)

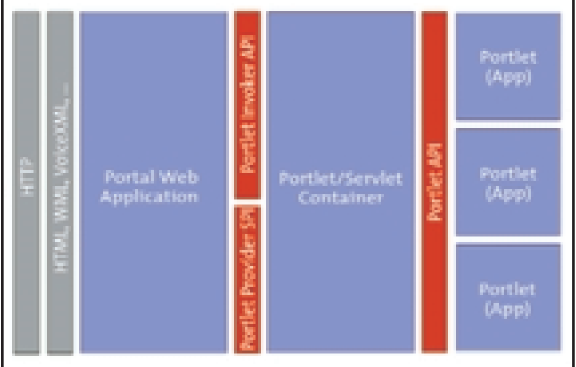


FIG. 7: PORTAL REFERENCE ARCHITECTURE

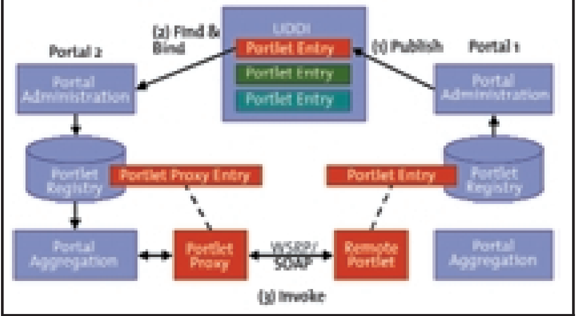


FIG. 8: PUBLISHING PORTLETS AS WSRP SERVICES AND BINDING WSRP SERVICES IN PORTALS

The Portlet API – WSRP Relationship

The portlet API and WSRP will be able to cooperate in a very beneficial way. WSRP services may be integrated in portals through portlet proxies written to the portlet API. Conversely, portlets may be wrapped in and published as WSRP services.

Figure 8 illustrates how one portal server publishes a portlet to a UDDI registry as a WSRP service so other portals can find and bind it. Typically, a portal has local portlets listed in a portlet registry. To make a portlet available as a WSRP service, the portal's administration UI may allow administrators to select a portlet and publish it to the UDDI directory. An adapter exposes the portlet externally by providing the SOAP operations defined in WSRP.

Once the portlet entry is listed in the UDDI directory, other portals can find and bind to the referenced WSRP service. To make a WSRP service available as a portlet, the portal's administration UI may create a portlet proxy entry in the local portlet registry with the information obtained from UDDI. Generic portlet proxies wrap the WSRP service.

Once the portlet proxy entry is in the local portlet registry, users may, for example, select it in the customizer and put instances of it on their pages. When a portlet proxy instance is invoked during page aggregation, the portlet proxy will marshal the method arguments in a SOAP request according to the WSRP standard and send it to the WSRP service. Then it receives the SOAP response from the WSRP service and unmarshals the result to provide it to the portal.

Conclusion

The Java Portlet API and WSRP standards will enable interoperability of portal servers, local portlets, and remote, interactive, user-facing Web services. The ultimate goal is a large number of portlets that can be run on any compliant portal server locally, as well as remote WSRP services that plug and play with all compliant portal servers. It's hoped that a portal component market will come into existence – one in which a large variety of application providers, ISVs, and the open-source community offer readily available portlets or visual Web services.

WebSphere Portal 4.1 already employs the concepts described in this article – an API for local deployment of portlets as well as a Web services interface for remote portlets. WebSphere Portal will support both the Java Portlet API and the WSRP standards, as soon as they are available, by upgrading the local and remote portlet interfaces available to the interface definitions in the emerging standards.

SITRAKA

WWW.SITRAKA.COM/PERFORMANCE/WDJ

A Conversation with Carol Jones

The chief architect of WebSphere Portal 4.1 talks about creating a platform for the future

Along with the official product name change to WebSphere Portal, there are significant changes in WebSphere Portal 4.1, the new release that Carol Jones, chief architect for WebSphere Portal recently described to Jack Martin, editor-in-chief of **WebSphere Developer's Journal.**

WSDJ: WHAT ARE THE IMPORTANT FEATURES ABOUT THIS NEW VERSION OF WEBSPHERE PORTAL?

Carol: When WebSphere Portal was first released in the early part of 2001 it had a focus on consumer Portals and pervasive Portals. In that first release we looked at how to take each Portal page to pervasive devices or wireless mobile phones.

In 2001 we started to look at enhancing the product to address requirements that our enterprise customers have and get better integration with the rest of the WebSphere family. We wanted to get deeply integrated with our application server and strengthen our capability in security and user management and all the features that you care about when you are building employee or supplier Portals that really carry your enterprise function.

On one level, the Portal server is about presentation on the "glass" – on the browser, on your mobile phone, on your PDA – whatever device you want. That's the big focus of the product and what we've done in WebSphere Portal 4.1 really strengthens the capabilities in all of those areas. You'll see a lot more depth in our page-layout capabilities, like the ability to rebrand the portal using skins and themes.

WSDJ: WHAT IS REQUIRED TO INSTALL WEBSPHERE PORTAL?

CJ: All the essentials are in the box, including the database and the application server, and all the related products. When you first install the portal server, you basically have a working portal right away. It seems pretty obvious that you'd want that, but this is really a different concept for IBM software. From that point, you go through a process of refining the portal by adding your own portlets, changing the look and the branding, and maybe adjusting the security and user management system to suit your specific needs.

WSDJ: LET'S TALK ABOUT THE PAGE-LAYOUT CONCEPT.

CJ: We're trying to make sure companies that deploy the Portal server aren't constrained by our ideas about the design. Our customers don't want an IBM look and feel. They want to put their image on the Portal server, to make it their own. Additionally, this has made it easier to change the language of the portal, which supports our international users and it helps with developing sites for people with special needs.

WSDJ: LIKE BUYING A HOUSE AND DECORATING IT TO YOUR TASTE?

CJ: Right A business partner showed me a picture of what they wanted and asked, do you think we can achieve this? Would that be a bad thing if we change your look out of the box to be so different? I said, "It would be the greatest compliment we could have." We don't want the Portal server to have some rigid look.

WSDJ: WHAT SKILLS ARE REQUIRED FOR THAT?

CJ: Basic Web design skills. It's helpful to understand cascading style sheets and to look at our JSP templates and change the fragments that control the masthead and things like that.

WSDJ: ARE YOU MOVING MORE TOWARDS A TYPE OF LAYOUT CONCEPT WHERE IF YOU HAVE BASIC GRAPHIC DESIGN SKILLS YOU CAN ACTUALLY START CHANGING THE WHOLE THING AROUND?

CJ: No. It deals more with Administrators. We want to enable companies with a hierarchical structure to manage their portals more effectively. You could enable the highest level administrators to define the basic structures of the page. Maybe those people have significant Web development skills. Not everyone has those skills so at the next level down, they might open up parts of the page and let administrators control the third column, say, or the middle column. They can add content or remove content or rearrange things. Basically, they can do whatever they need within some bounds. We're really interested in this idea of delegating administration tasks down through an organization.

The beauty of it all is that if your company merges with another company and you have a change in your design philosophy, and you want to rebrand all that stuff, you don't have millions of pages to update. You change things at the top level and the changes cascade down through the system.

In everything that I've talked about so far, the layout has to be a spatial view of the Web, but what we think is important about this system is that it expands in other spaces. It doesn't have to be a visual layout on a column-oriented browser. It could be a layout on a mobile phone that has simple pictures or menus. So maybe the device doesn't have all these visual considerations, but it does have different navigation concerns that need to be addressed. Maybe we're talking about a mobile device, a PDA, or even voice – all these structures have a logical equivalent in other markups.

WSDJ: CAN YOU GIVE ME A SUCCINCT DEFINITION OF A PORTLET?

CJ: The term portlet refers to a small, portal application, usually depicted as a small box on a Web page. Portlets are reusable components that provide access to applications or base content and other Web resources. For example, Web pages, Web services, applications, syndicated content, all those things that could be accessed through portlets. Any particular portlet is developed deployed, managed, and displayed independent of other portlets. The analogy would be that you think about a servlet as displaying a whole Web page. With a portlet, think about it as displaying one portion of a Web page. So a portlet is something that contributes to a bigger Web page.

WSDJ: CAN A PORTLET HAVE INFINITE BUSINESS LOGIC INSIDE OF IT?

CJ: It can be as simple or as complex as you like. The big principle of a portlet is that it is an independent, very small application – and application is a key word here. A lot of times people look at these Portals' UIs and they conclude that it's just static content or whatever. But the real purpose of a portlet is to be a whole application in itself. It might collect data from you, or it could do calculations or have some interaction with a back-end system that displays some answer and really have an entire interactive nature unto itself. But it's only taking up one small portion of the screen and it's sitting there side by side with other portlets.

WSDJ: CAN I USE WEB SERVICES IN THE PORTAL?

CJ: Of course, this is one of the most important new features in the release. Basically, you can take any portlet and publish it to a Web services directory. Then, from another portal you can find it in the directory and just bind it in. The portlet looks like it's running locally on the second server, but it's not. We communicate to it over SOAP, and it actually runs on the portal that it was published from. The cool thing is that no programming is required to do this. It's all done by administrators selecting portlets from lists and publishing them.

WSDJ: HOW DO YOU SEE THIS BEING APPLIED?

CJ: Web services support really resonates with people. You can have these ideas about federated Portals where portlets can be running either locally or remotely and can be shared back and forth very freely. The parties sharing the portlets might be different units of a company, or two different companies. Sharing portlets this ways lets them offer their applications and services in a way that is really easy to integrate into a portal. There's no programming or even web page development involved.

WSDJ: LET'S MOVE ON TO COLLABORATION.

CJ: The idea is to treat Portals as communities of users. It's not just connecting data and displaying data and the more technical concepts. It's about connecting people and helping them work more effectively. We have an edition called WebSphere Portal Extend. We've introduced something we call Collaborative Places. It's a shared area of the Portal that organizes your pages and your applications and keeps a membership list and shared bookmarks - anything that your group or your community of interest needs to do its work effectively. This is where we've focused a lot of effort integrating our Lotus products.

WSDJ: HOW WOULD A NOTES USER VIEW THIS NEW COLLABORATION FUNCTION THAT PORTAL OFFERS?

CJ: This is all browser-based access and we are backing it up with lower-level servers that access Notes documents, QuickPlace, Team Rooms, Discovery Server. All those functions that were available through Domino and other Lotus products are now available through the Portal and lower-level services.

WSDJ: ARE YOU SAYING PORTAL ELIMINATES THE NEED FOR THAT NOTES-SPECIFIC CLIENT TO BE ON THE MACHINE SO THAT YOU ACCESS NOTES AND DOMINO USING A BROWSER?

CJ: I wouldn't go quite that far – the clients are still important. You can already get browser-based access to Domino Services, QuickPlace, and Sametime services. The problem is they're all separate. What we're doing in the portal is integrating those into a richer, customizable user interface. For example, you can create a list in HTML of people and with a simple tag change make the list become active so that it automatically shows you if each person is available in Sametime or not. We're giving you the capability of more easily adding this kind of collaboration capability to ordinary portlets.

WSDJ: SO YOU'RE EXTENDING LOTUS TOO?

CJ: Right. One portlet might show some data out of certain Notes databases, side-by-side with another completely different Web application. It's really taking the functions that were available in a wide variety of Lotus products and making them available seamlessly with other Web-based portlets.

WSDJ: LET'S TALK ABOUT SYNDICATED CONTENT. FOUR YEARS AGO THAT WAS SUPPOSED TO BE ONE OF THE GIGANTIC PLAYS ON THE INTERNET. WITH THE EXCEPTION OF CNN AND YAHOO, THAT HASN'T GONE ANYWHERE AND AOL CHARGES THEIR SUPPLIERS THROUGH THE NOSE FOR IT OR MERGED WITH THEM. HOW DO YOU SEE THAT PLAYING OUT INSIDE A PORTAL PRODUCT?

CJ: Syndicated content is important, but not so much because of news and entertainment. What's more important about syndicated content is to apply the techniques and standards to a company's internal content. In Portal server we actually have built-in portlets that support some of the standard content formats. That means that you can set your portal up to behave the way Yellow Bricks or Yahoo! or some of those companies do, where the publishing and approval process is separate from the display. We call this "self-syndication".

WSDJ: LET'S SAY A COMPANY HAS 40,000 EMPLOYEES AND A MODERATE NUMBER OF INTERNAL ANNOUNCEMENTS AND PRESS RELEASES GOING OUT EVERY DAY. HOW WOULD THAT PROCESS WORK FOR THEM?

CJ: Assuming you've got content in standard format, we can already serve it up. The question is, "What's the bit of magic that gets the content in that format?" We've provided guides showing you how to take the popular content management products from Interwoven, Vignette, and other companies and use them to set up forms and the approval process, and then publish into formats that we already understand. Of course, our own tool is Web Content Publisher. It can actually put up a form that collects data in a standard format and puts it out in a way the Portal server already knows how to serve up.

WSDJ: HOW ARE YOU ADDRESSING SECURITY IN THE PORTAL?

CJ: First, you must manage the user population. How are you going to register users and store their profile information? Most companies already have a user population that they are managing in some directory. We can work with existing directories. We don't require you to make a new one, migrate everything over, clone it, or anything else. We have extensive capabilities for managing user profiles, called "WebSphere Member Services." We're turning that into a common component that we'll use in a lot of our products, which means different products in the WebSphere family can share the same user population. There may be a new user population that you built from scratch when people visited the Portal and registered, or it may be a directory or database that you already have. The system is configurable. Our whole design philosophy, is to make it as adaptable as possible. We know that people are not starting from scratch.

Now let's look at security. You've got all these users and they're registered. What do you do to ensure the privacy of their data and ensure the security of the system? You have to think about authentication (are you really who you say you are?), single sign-on (once you are logged in, we trust you, and you won't be prompted again), and authorization (what can you see and do in the portal?). Authorization is this idea that now that we know who you are, we want to build the page that you're going to see. All of that is driven by an internal authorization system.

WSDJ: HOW DOES THE AUTHORIZATION SYSTEM WORK?

The portal view that you experience as a user is entirely dependent on the access rights that you've been given. If an administrator says that you get to view the home page and these three portlets, then that's what you get to see. If another guy gets access to five more portlets, he sees eight portlets instead of three. That authorization system is key to this dynamic process of building up your page. It's a delegated concept so that access rights can change as you work down through the organization.

Maybe you have groups of managers that can see these things, but other employees can't. There's just a lot of flexibility in what you can grant access to. You can also take that idea and grant access to certain portlets for administration tasks. A simple example is, if I give everybody access to the news portlet, then everybody sees the news. But suppose I give some people access to the portlet that installs new portlets, or the portlet that clips Web pages, or the portlet that manages user groups. By granting access to those portlets, I have actually enabled that person to perform that piece of work. We've used the infrastructure to manage itself.

WSDJ: ISN'T THIS DECENTRALIZING THE ADMINISTRATION?

CJ: We're delegating for the appropriate reasons. We don't want to force people to do that if it isn't appropriate for their company.

WSDJ: IF I WERE USING PEOPLESOFT AS MY LEAD-TRACKING MECHANISM AND SAP TO KEEP TRACK OF EVERYTHING ELSE, DOES PORTAL OFFER ONE UNIFIED EXPERIENCE WHERE EVERYBODY SEES WHAT THEY NEED TO SEE, WHEN THEY NEED TO SEE IT?

CJ: Right. It's unified and rebrandable to fit your company. It's not SAP's, PeopleSoft's, or IBM's view of what you want to see.

We've also added the capability for Portlets to communicate with each other, we've got a technology preview of something called Click-to-Action. The idea is that portlets built by independent parties can now communicate with each other without having to use the low-level messaging APIs.

WSDJ: USING THE CLICK-TO-ACTION CONCEPT, IT REALLY DOESN'T MATTER WHAT'S ON THE BACK END.

CJ: The communication can happen between the two no matter what is behind it. As long as one can say, "I have employee numbers," and the other says, "I accept employee numbers," they can do the interchange pretty automatically. Also imagine where there is a person's name, we can have a little awareness indicator that tells us whether or not she is online. If we needed to verify something with that person, we could click on the name, send her e-mail or an instant message, whatever was appropriate, and get in touch with her. We could also find documents that she wrote – any kind of integration like that is possible. That's why this is much more than putting the UI together in a different way. It's much deeper than that.

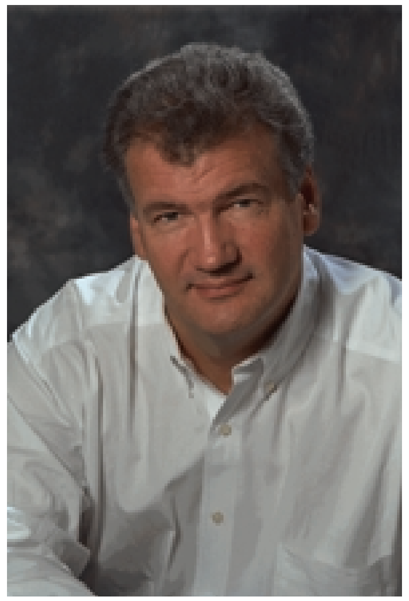
ABOUT THE AUTHOR

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

LOOKING AT IBM PORTAL WITH LARRY BOWDEN



Jack Martin, editor-in-chief of **WebSphere Developer's Journal,** *recently spoke with Larry Bowden, vice president of IBM Portal*

WSDJ: WHAT WAS THE GENESIS OF THE IBM PORTAL AND HOW WERE YOU INVOLVED?

Larry: We built portals long before they were called portals. Customers have been describing to IBM for several years how they wanted to use business technologies to lower costs and improve productivity for employees by bringing the applications, content, processes, and people interactions to a new Web-based workspace. Some customers focused on employees as their primary goal, others on partners, and still others on customers, but they had similar views of the values they wanted. About two years ago, the maturing technologies used to create portals and the market dynamics lined up, providing a potentially large opportunity for IBM, and it was an ideal time to put a specific initiative in place around the emerging portal market. My involvement was simple, I was asked to lead the portal initiative for IBM.

WSDJ: HOW DID IT GROW FROM THERE?

LB: The portal team had to look across the full breadth of technologies, products, services, sales, etc., available in the Software Group and outside IBM to construct a compete portal. IBM participated in essentially every major area important to a portal – authentication, security, Web application server, collaboration and knowledge management, enterprise content management, mobile and wireless, etc. We built a team that was fanatically focused on what the market needed and then looked across IBM Software Group and abstracted any asset, component, or product needed to construct a portal solution to match the market needs. We needed to create the heart of the portal offering, the portal framework. We have a powerful development leader and team that's delivered new framework advances at a record-breaking pace.

We use software from Tivoli, Lotus, WebSphere, DB2, and Pervasive Computing divisions in the portal solution. We used the WebSphere brand to name the product, but it required capabilities and competencies from every IBM software division. That is what's unique about portals and what gives IBM a decided advantage.

WSDJ: WHY DO YOU THINK THEY CHOSE YOU TO LEAD THE PORTAL EFFORT?

LB: During my career at IBM, I've had many years in software, I've had roles in development, marketing, manufacturing, research, planning, even in a sales office, and I've been a financial officer for a lab site. A spectrum of collected skills and experiences prepares you to handle whatever gets in the way of executing the mission. By design or happenstance, the breadth of my roles in IBM seemed to directly match what was needed to get a new IBM initiative off the ground and penetrating a new market. The IBM portal initiative has a distributed marketing plan, multisite development worldwide, dual sales team structure, converged services capability, multidiscipline technical support, a massive partnering program, and the portal product itself simplifies a complex set of technologies that could only have been done with an extremely dedicated and diverse team.

WSDJ: THAT WAS THE KEY, YOUR BROAD EXPERIENCES?

LB: I think it helped. If asked what to look for in a person to start another cross-discipline, cross functional initiative, I'd describe a person with a shallower but broader set of experiences versus a narrowly focused person.

It wasn't just me. We started the initiative with a core of six people, which surprises most IBM-ers because we have 100–200 times that now and close to 3,000 salespeople on portals. But the dynamics of the first six people was interesting. All the team members had to do more than one thing. On any day, depending on need, they'd have to be evangelists, work-with partners, make customer calls, define technical requirements, build and execute demand-generation plans, competitive analysis, sales collateral, education of partners – they did it all. Over the past two years we scaled up dramatically, and we now have a full complement of support teams in each critical area, but for a while, it was a small team determined to make a difference. I think they take pride in how their sacrifices of time, sweat, and brainpower have paid off. If you ask them, "What's been the 'funnest' project you've ever worked on?" I expect many will say this one is at the top of the fun meter. They really had and still have ownership to make it happen. All of them, greater than two years later, are still on the project.

WSDJ: WHAT HAVE BEEN THE KEYS TO SUCCESS?

LB: We've had six deliveries into the market. There was a pace we needed to work at to get out in front and be the leader, and we had to develop and deliver new functions quickly. Usually you don't see projects that have six deliveries in two years. It's almost too fast for customers to consume.

We have a development team in portals that's second to none. I'm not hyping them, I've spent 15 years in development and been involved in a lot of successful development projects, and this development team is exceptional. They've met every challenge and have a reputation of committing aggressive project plans and then delivering full function, on time, with quality. At the end of the day, once all the brochures are gone, and all the demos are retired, and the presentations are forgotten, it's the code that is the lasting value for the customer. That's where we focused – product value, and I was pleased to see the portal team stay focused on what was important to get to the market and not get caught up in the marketing hype of portals

The initial plan was designed for small, incremental increases in functionality at a very rapid pace. This delivered on two important success factors. In a quickly evolving market, predicting what requirement was going to emerge next was difficult. The rapid cycle times allowed us to respond to new requirements quicker than competitors. Also, we're very focused on delivering rock-solid software, and the intentional short development timeframes allowed us to put the portal code through multiple extensive system-test cycles and flush out any problems early in the product life.

WSDJ: DO YOU PLAN TO CONTINUE AT THIS PACE?

LB: We'll have more spacing between deliveries. We're adjusting to the market needs. Now that we have several system-test cycles completed and rich functionality, we'll change to delivery spacing so customers are doing fewer upgrades.

WSDJ: THIS SEEMS TO ME LIKE IBM IS OPERATING LIKE IT'S A SILICON VALLEY–TYPE COMPANY WHERE YOU LIVE ON MEAN AND LEAN, AND YOU'RE BENDING AROUND THE MARKETPLACE, SCOPING THE PRODUCT AROUND WHAT CUSTOMERS WANT, DISCARDING WHAT THEY DON'T. IS THAT ACCURATE?

LB: You very much described the design point. You can make a mistake in having a small set of customers overly influence your views. You miss important trends if you're not careful. In essence, based on the confusion in the market at the time, we selected customers that represented the heart of the market profiles.

The basic view was that the portal opportunity would be big, but it was too new an area for any real market-sizing estimates. There was no a clear definition of what the portal should do; everyone had a different definition of the portal. Until we got a better feel for what was needed, we structured the project for speed of response to what we learned and to not be too rigid in approaching the next delivery. Getting a lot of feedback from customers allowed us to tune our offerings to match customer demands.

It was encouraging that there were so many views on the role of a portal and the types of portals. These were great signs that the market was still early in the maturity cycle, the right phase for a major portal initiative. This is the type of environment where a company can quickly demonstrate leadership and bring order to the chaos.

WSDJ: IS IT ACCURATE TO SAY THAT SOFTWARE COMPANIES ARE MAKING SERIOUS INVESTMENTS IN PORTAL BY USING IT AS THEIR ENGINE? YOU'VE GOT A VERY STRONG THIRD-PARTY ENDORSEMENT FROM COMPANIES WITH THE TALENT, MONEY, AND BRAIN-POWER TO MAKE THEIR OWN VERSION, AND YOURS IS SO GOOD THEY'RE DECIDING TO LICENSE FROM YOU.

LB: Many companies have the ability and resources to build all kinds of software, but the question isn't capability. If a software development company had the resources to build a relational database, I wouldn't recommend building one; outstanding technology is available at great prices – look to the database experts. Companies want to leverage the values the portal provides without reinventing what's available from IBM. We can help them speed their solutions to market and stay focused on their unique value and core competencies that can differentiate them from their competition, and avoid wasting critical resources duplicating what IBM already offers. The key is the open standards–based approach to the portal architecture. It provides options.

We provide a complete, robust, scalable, horizontal portal that allows any application to benefit from that and be integrated for the end users. And the ISVs provide a diverse set of application components that make it easy for customers to adopt their applications. Customers and partners see WebSphere Portal as a logical choice.

WSDJ: YOU'RE PARTNERING WITH OTHER SOFTWARE VENDORS THAT SOMETIMES OFFER SIMILAR PRODUCTS TO IBM. WHY?

LB: It's about the best solution for the customer. Customers have an environment established to run their business. In that environment they've made strategic choices or are in the process of making choices and want the portal to fit into their existing environment and leverage these decisions. To provide a customer maximum value, you have to make sure you fit into their environment, integrate and leverage their previous investments, and add incremental value. We work with the major ISV partners in the industry. IBM's strategic alliances number over 70 and growing. We're systematically enabling partners with WebSphere Portal to make sure our customers not only have the full benefit of all the partners' capabilities through the portal, but to reduce the cost of integrating those applications.

WSDJ: IBM IS GOING TO START WINNING THE SOFTWARE GAME AGAIN?

LB: In the middleware business, there's no doubt that IBM is the market leader. Portals are the hottest thing happening in the middleware area, and we intend to be the dominant player.

WSDJ: WHAT ABOUT MICROSOFT?

LB: We haven't seen them in the enterprise portal space. We expect to see them in the small- and medium-size business segment, and they'll be the primary competitor for that segment.

I agree with the analysts that they have a long way to go to expand the document management–oriented solution they have. But bottom line, they have the resources to throw at their portal, so they're a viable threat in the SMB market. As you move to the enterprise, it's a different set of competitors. The companies with the wherewithal to play in the mission-critical middleware game will vie to become the dominant player.

WSDJ: DO YOU SUPPORT .NET AND JAVA WEB SERVICES ALSO?

LB: Our strategy is to have the portal be the single point of personalized interaction with the content, applications, processes, and people for the user. The portal needs to be able to seamlessly interact with Web services regardless of the source.

WSDJ: WHAT ARE THE THREE MOST SIGNIFICANT FEATURES IN THIS NEW VERSION?

LB: Completeness, because a customer doesn't want to worry about integrating lots of pieces. The portal takes a complex set of capabilities and simplifies it into a concise, manageable offering that delivers the superset of capabilities customers demand. That's number one.

Number two, a portal is useless without applications, processes, the people and everything available to it. The industry is building the integration components. New partners are committing to be integrated into the portal; this demonstrates that the industry is rallying around this environment. Therefore, customers don't have to do integration with various applications. It's already done, lowering their cost and all that goes along with it. You've got to make sure it's complete and when it's complete, don't make the customer spend time getting the rest of the components that they want.

Third, we've commercialized the ability to take anything, any portlet or application element inside the portal, and publish it as a Web service to a UDDI registry. We can also search a UDDI registry, bring in a Web service, convert it to a portlet, and use it inside the portal. This capability will open up the market so that when application providers create their applications and new functions are created by companies with Web services, you can depend on the portal to integrate those Web services for you.

WSDJ: HOW DO YOU ACCOMPLISH PEOPLE INTERACTION?

LB: There's undeniable differentiation for IBM within WebSphere Portal for the ability to collaborate with others, to set up chats, to invite people to join certain activities teamrooms, awareness of people being online/offline, project management, being able to disconnect from the portal and take a project with you and connect back into the portal with updates. We've leveraged the competencies of Lotus to go to a whole new level of collaborative integration in the portal. Collaboration is seamless in the portal, not a special area you have to go to for collaborative tools. Collaborative capabilities can be embedded into the portlets for seamless interaction with other people. It's an IBM exclusive and we're excited about its value.

WSDJ: WHEN I SPOKE TO CAROL JONES, WE TALKED ABOUT THE USER INTERFACE AND HOW THERE ARE NO CONSTRAINTS – THAT THE INTERFACE IS WHATEVER THE COMPANY WANTS IT TO BE, AND YOU CAN ACTUALLY HAVE AS MUCH CONTROL AS ALLOWED BY THE ADMINISTRATOR TO MAKE THAT INTERFACE. DO YOU ALSO SEE IT THAT WAY?

LB: A user doesn't even have to know how to spell the word "code" to personalize the portal. We're providing examples and predefined places so customers can see how the portal works, but our strategy is to deliver complete flexibility to users and administrators with no limitations. It's interesting to see how creative customers are in constructing diverse user experiences. They're doing cool stuff, and we don't want to constrain that creativity.

WSDJ: IS THERE RESISTANCE FROM IT TO PROVIDE THIS AMOUNT OF POWER TO THE USER?

LB: The portal is unique in that it can satisfy a company's IT objectives and the line of business (LOB) objectives. The infrastructure values help IT drive cost reductions, with managed security, single sign-on, user management, resource management, etc., from a centralization perspective. The flexible, customizable presentation capabilities support the line of business leaders, creating a very specific user experience for their constituency. The LOB is not stuck with a fixed user interface that they have to go back to IT and negotiate changes to if it doesn't meet their needs. They simply choose from the portfolio of portlets and change it. The LOB leaders have the ability to satisfy their unique needs, and the IT leaders can provide a more complete service to the LOB.

WSDJ: ARE THERE ANY NUMBERS THAT ARE QUOTABLE?

LB: We have customers who intend to come down on total cost of ownership (TCO) of computing expenses by 60% over three years. A large telecom company is justifying their actions based upon a corporate goal of reducing their total cost of computing. They looked at how much money they spent per person for the combination of hardware, software, servers, training, etc.; set a goal for the TCO; and are driving decisions to reach it. WebSphere Portal plays a major role in the infrastructure needed to drive such a significant TCO reduction.

WSDJ: HOW DO YOU GET THROUGH TO THE PARTNERS?

LB: The unique thing about a portal is that it's a channel for partners to expose capabilities to the IBM customer set. When you explain the values and capabilities of WebSphere Portal, a good portion of that value is the preintegrated applications available from partners. We want to communicate the available portlets from partners. The availability of more preintegrated portlets clearly indicates lower integration costs for the customer. Every partner wants IBM to put their product in front of our customers. Wouldn't you like to have the IBM sales force including your product when talking to the best and biggest companies on the planet, and communicating that you and IBM have already done the integration work?

There's an entire network of implementers for the portal. They don't have a specific piece of software; they're trained on how to install, configure, and tune the solution for the best operation system. We help them build demand; we have comarketing activities to attract potential customers that need their domain expertise.

WSDJ: HOW DOES A PARTNER ENGAGE IBM TO INTEGRATE THEIR APPLICATIONS WITH WEBSPHERE PORTAL?

LB: There's a Web site (www-3.ibm.com/software/webservers/portal/partners/process) that contains integration information. They can download a contract from IBM and have all the supporting materials to integrate portlets and make them available to our joint customers. They can do this online; it's straightforward and self-service oriented. You want integration to be quick. You want the focus to be on new value for the customer and driving business together. The portlet program is an effectively running system and a key part of the overall growth plans for WebSphere Portal.

Web Services
Java XML
.NET Wireless

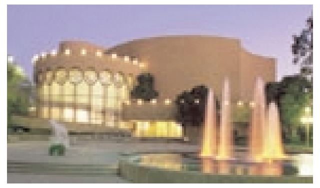
web services EDGE conference&expo | JDJ EDGE conference&expo | XML EDGE conference&expo | .NET EDGE conference&expo | wireless EDGE conference&expo

REGISTER NOW!
WWW.SYS-CON.COM

Focus on Web Services
Those companies that get an early jump on Web services to integrate applications for the enterprise will be the winners in today's challenging landscape. Take three days to jump start your Web services education!

Focus on Java
Hear from the leading minds in Java how this essential technology offers robust solutions to *i*-technology professionals and senior IT strategy decision-makers.

Focus on XML
Here's your chance to hear up-to-the-minute developments in standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.



OCTOBER 1–3, 2002
SAN JOSE MCENERY CONVENTION CENTER
SAN JOSE, CA

OWNED BY SYS-CON MEDIA
PRODUCED BY SYS-CON EVENTS

The Largest Web Services, Java, XML, .NET and Wireless Conference and Expo

The Conference...
Organized into four comprehensive tracks:
Web Services, Java, XML and IT Strategy
- Over 48 Information-packed Sessions, Case Studies and tutorials, plus 45 wireless sessions
- Thought leaders and innovators presenting the topics you most want to hear

Free...
FREE Web Services Tutorial to the First 100 to Register.
Attend a FREE Web Services Tutorial on October 3.
Register by August 15th to reserve your seat!

Who Should Attend...
- Develpers, Programmers, Engineers
- *i*-Technology Professionals
- Senior Business Management
- C Level Executives
- Analysts, Consultants

The Largest Expo...
PLATINUM SPONSOR
IONA
GOLD SPONSORS
ADOBE bea ORACLE VIGNETTE
SILVER SPONSOR
SilverStream
CORPORATE SPONSOR
COREL
MEDIA SPONSORS
JAVA DEVELOPER'S JOURNAL WebServices JOURNAL XML-JOURNAL COLDFUSION Developer's Journal wireless BUSINESS & TECHNOLOGY .NET Developer's Journal BEA WebLogic DEVELOPER'S JOURNAL WebSphere DEVELOPER'S JOURNAL

Powerful portlets enhance Version 4.1

WebSphere Portal Administration

BY CHRIS PAUL

One of the most significant sets of enhancements introduced in WebSphere Portal v4.1 can be found in portal administration. The improvements offer portal administrators a wider set of functional capability, improved usability, and a robust delegated administration facility.

ABOUT THE AUTHOR
Chris Paul is the design lead and manager of IBM's Pervasive Computing User Experience Design Group, an eclectic blend of artists, designers, and engineers. He previously led IBM's Web Analytics UI Design and Development team and was also the lead UI designer on WebSphere Studio. He holds an MFA in graphic design from the Yale School of Art.

E-MAIL
chrispaul@mind-spring.com

Providing this wider functionality are new administrative portlets that have been added to what was previously available. Highlights include portlets for installing and adding other portlets to the portal's registry; portlets for managing users and user groups; an improved portlet for creating and managing access control lists; and portlets for clipping Web pages, publishing Web services, setting portal-wide settings, managing logs, and other common administrative tasks.

As in previous releases, the portal is administered through the portal itself – v4.1 introduces the Portal Administration page group. This page group organizes all the administrative portlets into one convenient access point for administrators. The various portlets are organized into the following categories: Portlets, Portal Settings, Users and Groups, Security, and Portal Content.

A Brief Overview of Administration Portlets

Figure 1 shows the WebSphere Portal v4.1 portal administration (showing the Portal Settings page with the Global Settings portlet active).

PORTLETS

Install Portlet

The Install Portlet allows you to install a local Web Archive (.war) file, register the new portlet with the current portal's portlet registry, and automatically activate the portlets, making them available for immediate use, all in one step. Use the Access Control portlet after installing new portlets to grant other users permission to use the new portlet resources.

Manage Portlet Applications

This portlet allows you to update, copy, rename, activate/deactivate, uninstall, view details, and configure the existing portlet applications.

Manage Portlets

This portlet provides the means to copy, activate/deactivate, rename, view details, delete, and configure existing portlets in the portal's portlet registry. You can view a list of all installed portlets or use the inline search functionality to search the portal's portlet registry and pinpoint the desired portlet.

Web Clipping

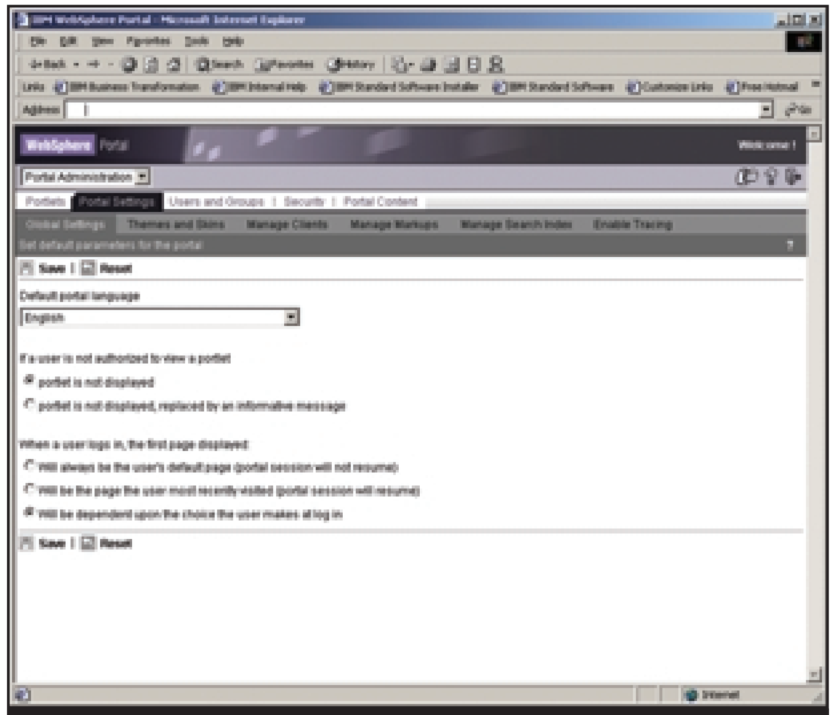
One of the most powerful new portlets is the Web Clipping portlet. This portlet is used to display sections of existing Web pages as portlets. You use the Web Clipping portlet to define a Web clipping. When you define a clipping, you can visually select portions of the page or clip all the text between specific tags. This way, you control precisely what markup is used to create your new portlet. The Web Clipping portlet can also rewrite the links inside the clipped page if you desire, so existing Web pages can be viewed without leaving the portal's own navigation structure. When you finish your Web clipping, a new portlet is created in the portal's registry. Whenever the new portlet is displayed on a portal page, it will retrieve the current version of the Web page and extract and display the specific portion you clipped.

Some sites you clip may require authentication to access the site content. The Web Clipping portlet provides options for no security, basic authentication, or form-based authentication. If the content to be clipped is under security, the administrator defining the clipped portlet must provide the appropriate credentials to access the content, then the appropriate authentication credentials must again be provided by the end user at runtime.

Web Services and Manage Web Services

WebSphere Portal provides extensive support for Web services. Portal administrators can publish and bind remote portlets as Web services, making them dynamically available in the portal's registry and seamlessly available to end users on demand.

Completing the publishing step places an entry for the portlet into the desired UDDI directory. The administrator of another portal can browse the UDDI directory to find previously published portlets and bind them into their local portal. This makes the remote portlet available as if it were locally installed; however, the portlet is actually running on the original portal server that published it.


IBM WebSphere Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer IBM Standard Software Customize Links Free Hotmail
Address Go
WebSphere Portal
Welcome !
Portal Administration
Portlets Portal Settings Users and Groups Security Portal Content
Global Settings Themes and Skins Manage Clients Manage Markups Manage Search Index Enable Tracing
Internet

Set default parameters for the portal

Save | Reset

Default portal language: English

If a user is not authorized to view a portlet
[✓ portlet is not displayed]
portlet is not displayed, replaced by an informative message

When a user logs in, the first page displayed
Will always be the user's default page (portal session will not resume)
Will be the page the user most recently visited (portal session will resume)
[✓ Will be dependent upon the choice the user makes at log in]

Save | Reset

FIG. 1: WEBSPHERE PORTAL V4.1 ADMINISTRATION

PORTAL SETTINGS

Global Settings

In the Global Settings portlet, you can change portlet settings such as the default language, the cache timeout values, and so on.

This version of WebSphere Portal also features settings that control how new user sessions are handled. Users logging in to the portal may wish to automatically return to the last page they viewed before logging off, so there's a setting to retain the state of the last visit. As the administrator, you can also decide what to display when a user tries to access a portlet without authorization. Unauthorized access can be ignored (the portlet is not displayed to the unauthorized user), or the portlet can be replaced by a configurable informative message so the user can take the actions necessary to correct the situation.

Themes and Skins

Themes provide the means for altering the visual aspects of the portal. WebSphere Portal Server uses a combination of JavaServer Page templates, cascading style sheets, and images to define the look and navigational structure of the portal pages. All these elements combine to form a theme. You can use themes to dramatically alter the appearance of the portal by adding your company logo, altering the color scheme, or changing the visual style. Themes are set at the page group level. Skins are basically the decorations rendered around the portlet when it's displayed on a page.

Using the Themes and Skins portlet, you can add, edit, delete, and choose a default theme for the portal. This portlet also provides the means to add, delete, and set a default skin for the portal. WebSphere Portal v4.1 allows any number of skins to be associated with a theme. When a user chooses a theme for a page group, you can control the list of available skins for use in conjunction with the theme.

Manage Clients and Manage Markups

The portal server supports several different markup languages so portlets can render themselves for a variety of desktop and mobile browsers.

Out-of-the-box, the page aggregation subsystem supports several markup languages and recognizes particular browsers and mobile-device user-agent signatures. The framework for supporting markup languages is open and extensible, so it's easy to support additional markups or new devices.

To support new browsers and devices, you add new markups and clients using the corresponding administration portlets. In the Markups portlet, the markup name indicates the folders used to store the page templates and the theme or skin files matching that markup language.

To add a markup, create a new entry specifying the MIME type and character set associated with that Markup. You also need to add all JSP templates associated with supporting a markup, such as new layouts, screens, skins, and stylesheets.

When the portal server receives an HTTP request, it matches the values in the user agent header against known patterns that identify common browsers for desktops, mobile phones, and other devices. Entries for these and other common clients are already set up, but you can add new ones using the Manage Clients portlet.

Manage Search Index

WebSphere Portal provides integrated text search capabilities, including a search portlet, a crawler, and a document indexer. The search service can search the portal's document repository as well as Internet content.

To prepare for searching, the search engine builds a full-text index to search documents stored in the local file system. The index can be compressed, and the size can be controlled for situations in which the index size needs to be limited. Use this portlet for creating, updating, and managing the search index.

Enable Tracing

Administrators can control the tracing and logging activity through the Enable Tracing portlet and also by modifying the configuration properties files of the logging subsystem.

The portal server records user activity in logs that can be processed by IBM Tivoli Web Site Analyzer. Overall usage statistics such as logins

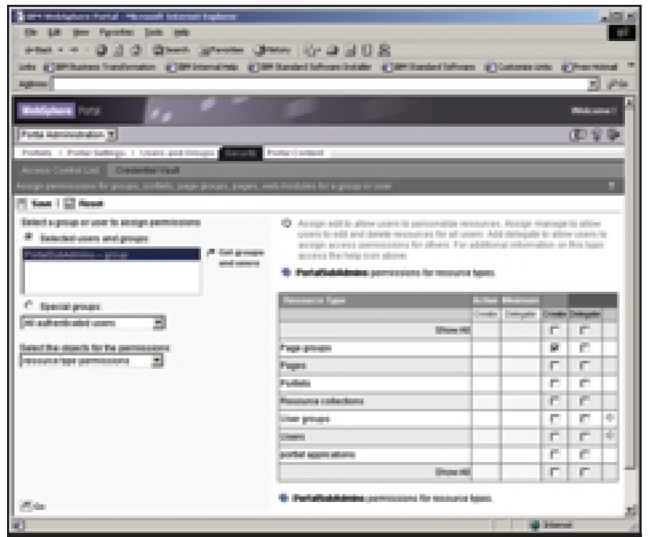


FIG. 2: CREATE PERMISSION

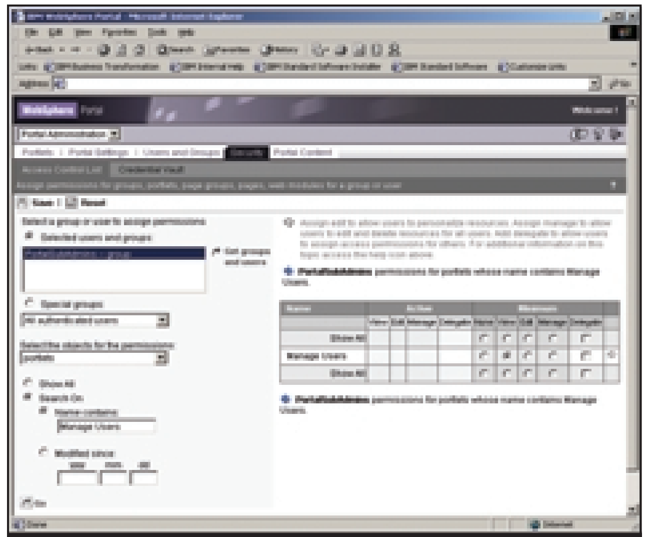


FIG. 3: VIEW PERMISSION

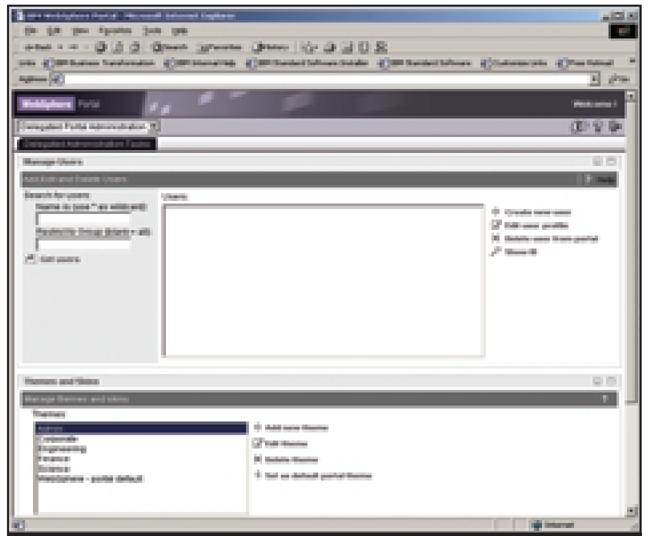


FIG. 4: CUSTOM PAGE FOR SUBADMINISTRATOR

and logouts, enrollments, and error conditions are tracked. Portlet and page usage statistics, including portlet actions, number of views, modifications, etc., are also tracked.

USERS AND GROUPS

Manage Users

In v4.1 portlets have been added that allow you to manage user and group information without leaving the portal. The Manage Users portlet lets you add users to the portal, edit existing user profiles, delete users, and view user IDs. Search capability has been provided to enable administrators to quickly pinpoint specific users.

Manage User Groups

The Manage User Groups portlet allows you to add user groups, delete user groups, show user group IDs, and manage user group memberships. As with the Manage Users portlet, search capability is also provided.

SECURITY

Access Control

The portal server enforces access control to portal resources, including pages, portlets, page groups, and user groups. After determining the user's identity, the portal server consults locally cached access control lists to determine which pages and portlets a user has permission to access. These access permissions are maintained using the Access Control administration portlet and are stored in the portal's administration database. Use this portlet to grant view, edit, and manage permissions to individual users or groups of users, so they may access specific portal resources such as portlets, pages, or page groups. Users may also delegate the permissions they hold to other users.

Credential Vault

Many portlets need to access remote applications that require some form of user authentication. For accessing applications outside the portal's realm, portal server provides a credential vault service that portlets can use to store user IDs and passwords (or other credentials) for a user login to a remote application. Portlets can use these on behalf of the user to access remote systems and achieve single sign–on authentication capability for portal users. The credential vault supports either local database storage or IBM Tivoli's Access Manager for secure storage and retrieval of credentials.

PORTAL CONTENT

The portal server includes a Content Organizer portlet that enables portal users to contribute and share documents. The Content Organizer portlet provides a workspace for storing, navigation, viewing, and searching portal documents and other content. The organizer is pre-configured to work with files and Rich Site Summary (RSS) formats. Additional content types, formats, and back-end systems can be integrated easily.

Hints and Tips

TYPES OF PERMISSIONS

There are two basic types of permissions that can be set in the Access Control portlet:

1. ***Permission on a specific resource*** (i.e. weather portlet, Joe's Home Page, My Company page group, PortalSubAdmins user group, etc.)
2. ***Permission on a resource type*** (i.e. portlet, page, page group, etc.): Used primarily to define which users and user groups can create new instances of a specific resource type. For example, to allow PortalSubAdmins to create page groups, the portal administrator would need to grant this user group CREATE permission on the resource type Page groups.

Figure 2 shows the CREATE permission granted to the user group PortalSubAdmins for the resource type Page groups.

DELEGATED ADMINISTRATION

As with any resource within the portal, access to administrative tasks can be controlled and fine-tuned by the portal administrator. The Access Control portlet (found on the Security page within the Portal Administration page group) provides the capability to grant any portal user or user group access to a specific administration portlet. Thus, administrators have the ability to delegate specific administrative tasks to other portal users or user groups in the same manner they give them access to view, edit, or manage more traditional portal resources, such as pages and content portlets.

It's important to note that to complete an administrative task, additional permissions are often required. For example, if a portal administrator wants to give a user group the ability to create users, he or she would need to grant the user group VIEW access to the Manage Users portlet, and to the page and the corresponding page group containing that portlet (such as the Users and Groups page in the Portal Administration page group), and CREATE permission on the Users resource type. VIEW access on the portlet, page, and page group ensures that the user group will see the portlet in their portal view and thus be able to interact with it. CREATE permission tells the portal that this particular user group has the ability to create new instances of the type Users. These permissions can be granted and controlled using the Access Control portlet.

Figure 3 shows the VIEW permission granted on the Manage Users portlet to the user group, PortalSubAdmins.

As a general rule, to modify a resource in the portal a user needs two distinct types of permission – an appropriate level of access to the specific resource and VIEW access to the "tool" or administration portlet to perform the modification. The administration portlets are the tools to manipulate various resources central to the portal's operation, such as users, user groups, content portlets, Web services, global settings, etc.

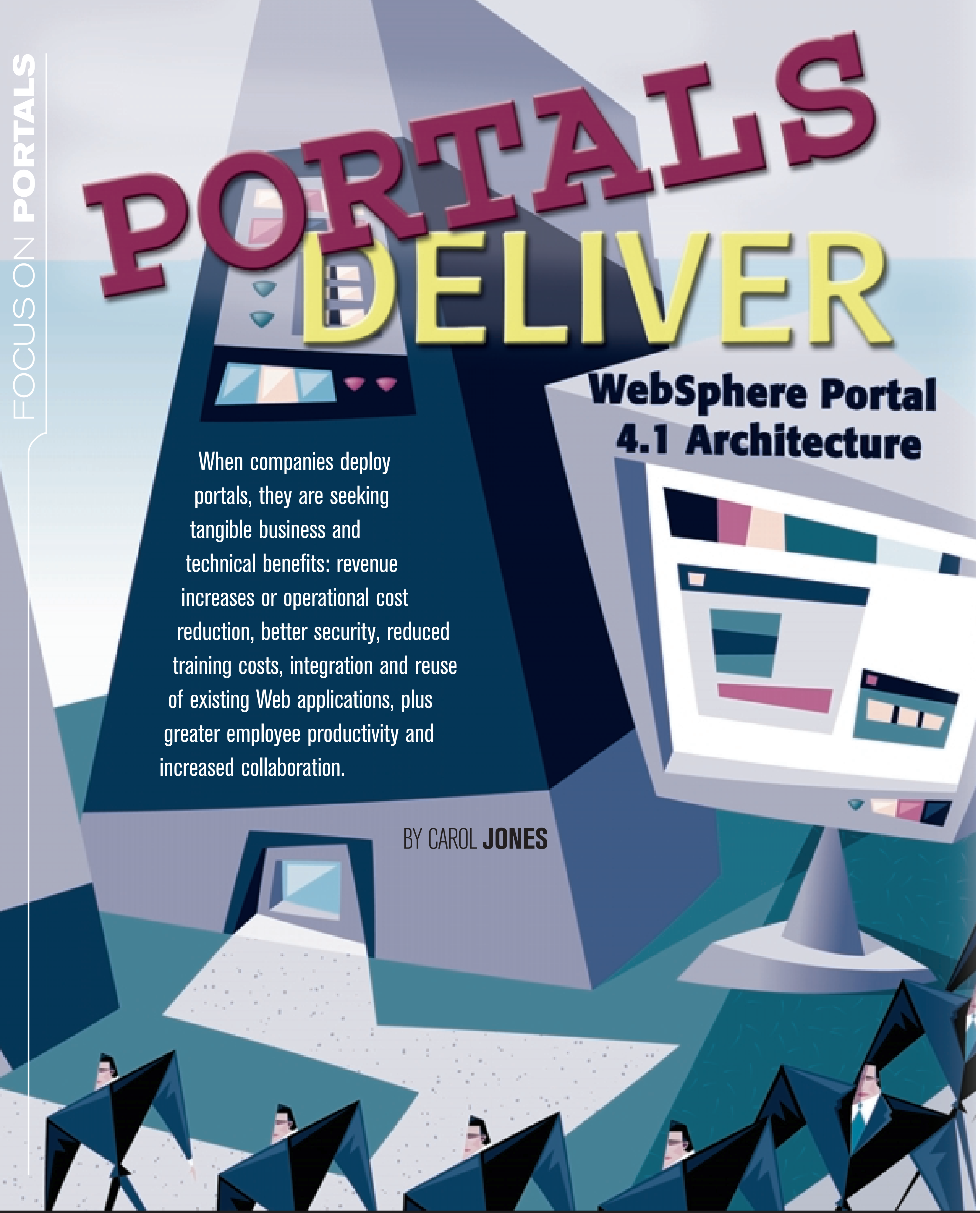
To fine-tune a user's ability to complete portal administrative tasks, the portal administrator can customize pages for various levels within the organization and reveal only appropriate tasks and portlets. Figure 4 shows an example of a custom page developed for a subadministrator of the portal.

USER GROUPS

The portal server uses group membership information to determine what page groups, pages, and portlets a user is authorized to view and edit. Users can be members of one or more groups, and groups may be nested (contain other groups). Nested groups inherit the access permissions set of their parent(s). A user is allowed access to portal resources when a minimum of VIEW access is granted for that resource to any group the user belongs to. Access rights can also be granted to specific individuals, but most companies find that it's easier to manage the access rights of groups instead.

PROLIFICS

WWW.PROLIFICS.COM



PORTALS DELIVER

WebSphere Portal 4.1 Architecture

When companies deploy portals, they are seeking tangible business and technical benefits: revenue increases or operational cost reduction, better security, reduced training costs, integration and reuse of existing Web applications, plus greater employee productivity and increased collaboration.

BY CAROL JONES

Portals deliver these benefits by providing a single access point to applications, consistent operational infrastructure, personalized access to relevant information, single sign-on, common presentation and a consistent user interface, and multidevice access.

A typical enterprise portal presents and unifies information from various sources such as productivity applications, databases, transaction systems, syndicated content providers, and existing Web sites. The portal combines independent, small applications into complex pages and presents them in a compact, consistent form. A typical enterprise portal is shown in Figure 1.

The term *portlet* refers to a small portal application, usually depicted as a small box in a Web page. Portlets are the component model for plugging applications into the portal. Any kind of Web content, such as Web pages, Web services, applications, and syndicated content feeds, can be accessed through portlets. Any particular portlet is developed, deployed, managed, and displayed independently of other portlets. Administrators and end users create personalized portal pages by choosing and arranging portlets.

Architecture

The single most important architectural concept of WebSphere Portal is this: it manages and displays portlet applications and Web pages that are unique to each user. The portal is not simply a static display arranged neatly into rows and columns; rather, it provides a role-based, personalized display. It combines content and applications into displays that may not have been designed or even anticipated by the providers of each content element. This flexible page-construction concept produces very dynamic displays that take into account the user's role, preferences, communities, and access modes through both desktop and mobile devices.

Figure 2 shows the WebSphere Portal architecture. In this article, we'll examine each of the main components.

SECURITY AND MEMBER SERVICES

Block 1 of Figure 2 shows the portal's security and user management services; more detail is shown in Figure 3.

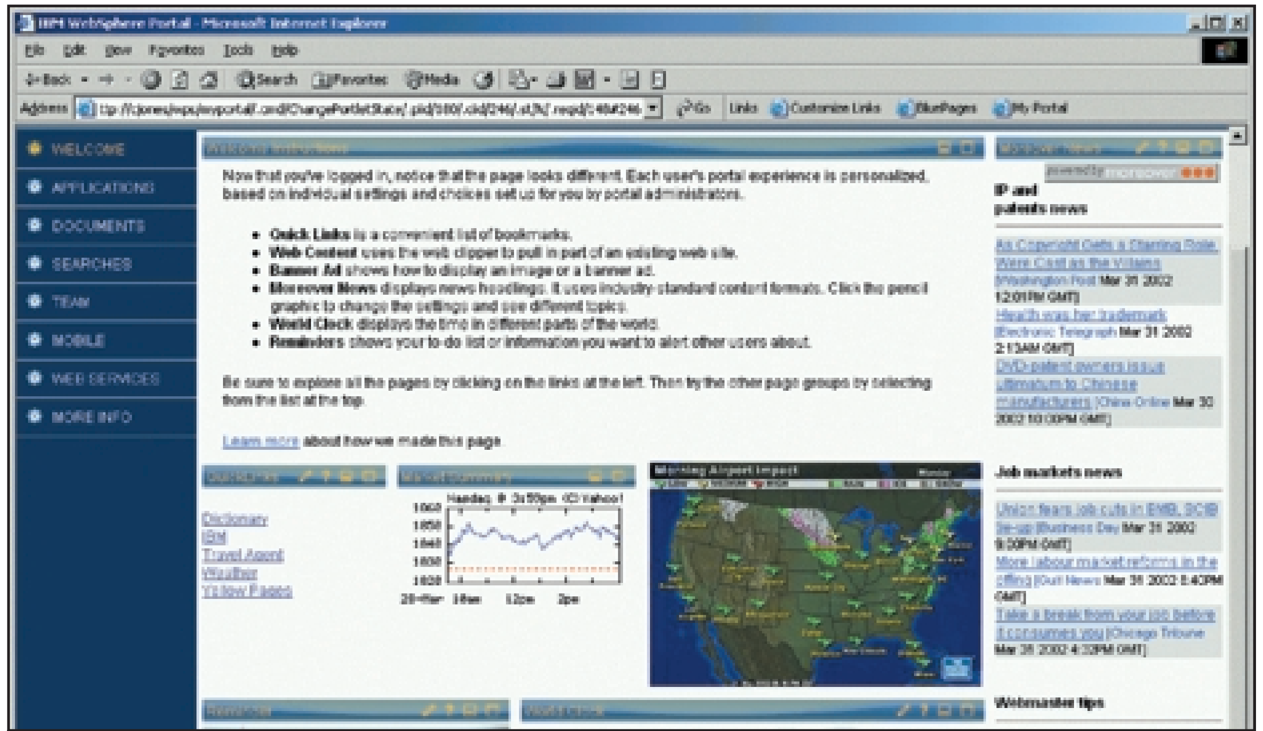
The component for defining portal users and user groups is known as WebSphere Member Services. The subsystem includes Web pages with which users can register and manage their own account information; administration portlets for managing user accounts and group information; plus a repository that stores all the information about portal users. It provides services to create, read, update, and delete users or groups in the repository. User profile information includes general information such as a user's name and user ID, plus preference information such as news topics of interest, preferred language, and interests. A user may be a member of one or more groups, and groups can contain other groups.

WebSphere Member Services uses a combination of a Lightweight Directory Access Protocol (LDAP) directory and a relational database to store user information. The database may be either DB2 or Oracle. Any one of several LDAP directory products are supported, including the Netscape (iPlanet) Directory Server, Microsoft Active Directory, Lotus Domino Name and Address Book, and IBM's SecureWay Directory Server.

A database is also used to store all the definitions that are needed for running the portal. These definitions include portlet descriptors, portlet settings and configuration data, page descriptors, and information about devices and markups used during page aggregation.

AUTHORIZATION

The authorization component controls access to all sensitive portal resources, including pages, portlets, page


IBM WebSphere Portal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links Customize Links BluePages My Portal
WELCOME
APPLICATIONS
DOCUMENTS
SEARCHES
TEAM
MOBILE
WEB SERVICES
MORE INFO

Now that you've logged in, notice that the page looks different. Each user's portal experience is personalized based on individual settings and choices set up for you by portal administrators.

- **Quick Links** is a convenient list of bookmarks.
- **Web Content** uses the web clipper to pull in part of an existing web site.
- **Banner Ad** shows how to display an image or a banner ad.
- **Moreover News** displays news headlines. It uses industry-standard content formats. Click the pencil graphic to change the settings and see different topics.
- **World Clock** displays the time in different parts of the world.
- **Reminders** shows your to-do list or information you want to alert other users about.

Be sure to explore all the pages by clicking on the links at the left. Then try the other page groups by selecting from the list at the top.

Learn more about how we made this page.

Dictionary
IBM
Travel Agent
Weather
Yellow Pages

IP and patents news

Job markets news

Webmaster tips

FIG. 1: A TYPICAL ENTERPRISE PORTAL

ABOUT THE AUTHOR
Carol Jones, a distinguished engineer at IBM, is the chief architect for WebSphere Portal.

E-MAIL
jcarol@us.ibm.com

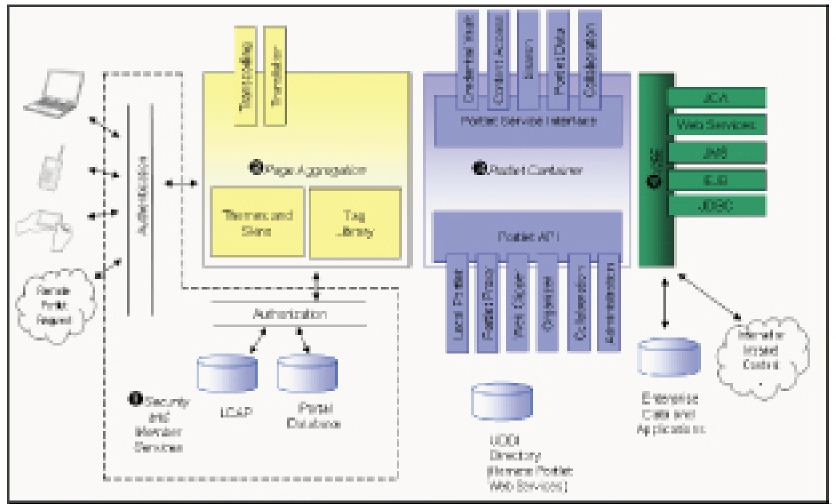


FIG. 2: OVERVIEW OF THE WEBSPHERE PORTAL ARCHITECTURE

groups, and other portal resources. It provides fine-grained access control and permissions, which can be delegated to other users.

Each user's personalized page can have a different set of portlets. The portlets on a page can be selected by end users or by administrators, depending on their access rights for the page. Administrators can specify that certain portlets are required, so end users cannot remove or rearrange the portlets.

AUTHENTICATION

Authentication is the process of establishing a user's identity. In the default configuration, WebSphere Portal uses form-based authentication, which means that the user is prompted through an HTML form for the user ID and password when trying to access the portal. The portal server requests that the application server validate the authentication information an LDAP user registry.

The WebSphere Portal authentication subsystem also supports the use of external authentication proxy servers. This is the most common scenario in production environments. When an external authentication proxy server is used, trust needs to be established between that proxy and WebSphere Application Server. The WebSphere architecture for doing this is called the Trust Association Interceptor (TAI). A TAI module converts security information specific to the authentication proxy into a format that can be handled by the application server. The exact details are different for each proxy server product. TAI modules for IBM Tivoli Access Manager and Netegrity SiteMinder are packaged with the portal server (all editions). The WebSphere Application Server documentation includes instructions for creating custom TAI modules.

PAGE AGGREGATION

Block 2 of Figure 2 shows the page aggregation system of the portal. Any complex row or column layout can be achieved with the portal server's aggregation and page layout system. Administrators at different levels of the organization can lock the layout or content of any area of the page. This way, a higher-level administrator can set up the basic structure of the page and fix certain portions or leave them open to modification by other administrators or end users.

Related pages in the portal are organized into places. Each can have its own color themes, skins, and page layouts. Themes are used to define the fonts, colors, spacing, and other visual elements; themes consist of cascading style sheets, JSP files, and images. Skins are the decorations and controls placed around portlets, such as title bars, borders, shadows, etc. Since the look and feel of each place can be completely different, places can be used to create multiple virtual portals running on one portal server. The entire look and feel of the portal can be quickly and easily changed by changing the themes and skins.

When the portal server receives an HTTP request, it matches the values in the user agent header against known patterns identifying common browsers for desktops, mobile phones, and other devices. Each major type of browser has its own page aggregation module that generates the appropriate markup for that device. The aggregation system is based on JavaServer Pages templates, and is completely responsible for rendering the requested page. Individual JSP templates compose each part of the display, such as page navigation links, columns, rows, title bars, and eventually, the actual portlets (see Figure 4).

Each portlet is responsible for rendering itself in the appropriate markup language, if possible. Some portlets may be available for all the supported devices, while others may be available only on a single device. Typically, portlets implement device-specific renderings by using a different JSP for each device. Thus, the user interface design of each portlet varies from device to device, so the user's experience can be optimized. A user's home page and individual portlets might appear very different on a mobile phone from how they would appear in a desktop browser (see Figure 5).

TAG LIBRARIES

The portal server provides a JSP tag library for easy access to portal functions. There are tags that return data about the portal or check for certain conditions. For example, you may want to display the logged-in user's name or find the path to the current theme folder.

For globally accessible portals, the portal server will search for and select the proper JSP, based on the target browser and target browser's settings for language and country. The portal server searches for the JSP for a portal from the most specific to the least specific.

The portal server's JSP tag library also provides tags for identifying and separating translatable text, so you don't have to create and maintain separate JSP files for each language. Using this technique, you could create one JSP view and when the page is processed, the strings are substituted using standard Java resource bundles. Different techniques are useful in different situations.

PORTLET CONTAINER

The portlet container, shown in Block 3 of Figure 2, provides the runtime environment for portlets. Since portlets are an extension of servlets, the portlet container uses the WebSphere Application Server's servlet container and adds capabilities that are specific to portlets. These additional capabilities include window event handling, inter-portlet messaging, access to portlet data, and the ability to dynamically discover portlet services.

Portlets inherit from HttpServlet. All the concepts you're already familiar with for servlet programming are directly applicable to portlets. The Portlet API is an extension of the servlet API, with additional methods to support the modes, states, events, and messaging capabilities that portlets require. It's important to remember that portlets are much more than static, visual displays or simple JSP views. A portlet is a complete application, with application and view logic as complex or simple as needed. The main conceptual difference between portlets and servlets is that portlets are always assembled into a larger portal page. Multiple occurrences of the same portlet display different data for each user.

Local portlets run on the portal server itself. They're deployed by installing Web Archive files on portal servers and are invoked by the portlet container directly through local method calls. Since local portlets run directly on the portal server, they provide the fastest performance.

The portlet container provides an interface to dynamically load portal services. This is important because it makes the services available to all portlets without requiring the service code to be packaged with the portlet. The implementation of such portal services can be exchanged or enhanced transparently without affecting the portlet. WebSphere Portal provides discoverable services for its credential vault, for managing persistent TCP/IP connections, and for managing the portal's content repository. Portal developers can implement additional services, such as search, notification, content access, or mail services.

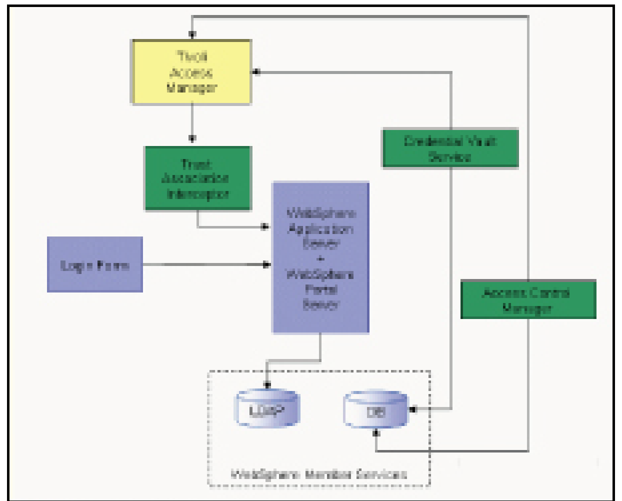
J2EE SERVICES

Portlets can use the complete spectrum of J2EE APIs and services to accomplish their application logic. For example, they might use Java Connectors to access enterprise applications, Web services, Enterprise JavaBeans, and so forth. Block 4 of Figure 2 shows some examples of the services that portlets might use.

Portal Services

SINGLE SIGN-ON

The portal server provides comprehensive single sign-on (SSO) support. Users want to be able to log on only once, and be known to the different parts of the portal without being asked to provide a different user ID and password each time they access different portal applications.


Tivoli Access Manager
Trust Association Interceptor
Credential Vault Service
Login Form
WebSphere Application Server + WebSphere Portal Server
Access Control Manager
LDAP
DB
WebSphere Member Services
FIG. 3: PORTAL SECURITY SERVICES

WebSphere Application Server uses Lightweight Third-Party Authentication (LTPA) tokens to provide single sign-on capability. When a user is authenticated, the portal server creates an LTPA single sign-on cookie containing the authenticated user credential. This encrypted cookie conforms to the format used by WebSphere Application Server and can be decrypted by all application servers in the shared domain, provided they all have the same cipher key. This cookie enables all servers in the cluster to access the user's credentials without additional prompting, resulting in a seamless single sign-on experience for the user.

Credential Vault

Many portlets need to access remote applications outside the portal's SSO realm. For these cases, portlets can use the credential vault service to retrieve user credentials for a particular application and to log in on behalf of the user. The credential vault supports either local database storage or IBM's Tivoli Access Manager for secure storage and retrieval of credentials. Portlets can use the credential vault server to perform the authentication using basic authorization, SSL client authentication, digest authentication, or LTPA.

Portlets that depend on remote connections require some way to maintain that connection as users navigate through the portal. The portal provides a persistent back-end connection service that maintains TCP/IP connections across page changes. It handles forms-based logins and stores the cookies that are returned as a result. For subsequent calls, the portlet can then ask the credential for an authenticated connection. This gives an HTTP connection with these cookies already set in the header. This way, portlets can maintain persistent, secure back-end connections.

Search and Content Management

WebSphere Portal provides an integrated search engine for searching local portal content as well as external content on the Web. The search engine supports free-text queries, with query assistance and query word completion. It can search documents in any language, and it also supports synonyms and stop-word lists and provides both document summaries and search-results clustering.

Content management systems define procedures for editing, managing, and publishing content for the portal. The portal server supports standard content formats, including Rich Site Summary (RSS) and Open Content Syndication (OCS) channels; these formats can easily be displayed by the portal server's built-in RSS portlet. This is just one example; other content formats can easily be displayed by other portlets. Additional content types, formats, and back-end systems can be integrated easily.

Portal Administration

Most portal administration is done through the portal itself, using administration portlets. Administration policies can be centralized or delegated through various levels of an organization. Granting access rights to specific

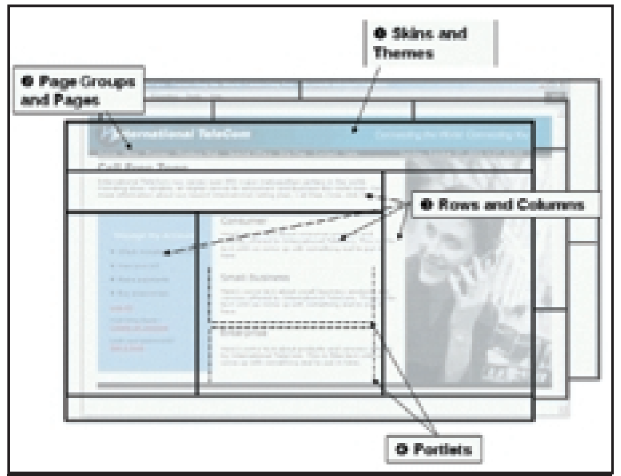


FIG. 4: THE PAGE LAYOUT SYSTEM

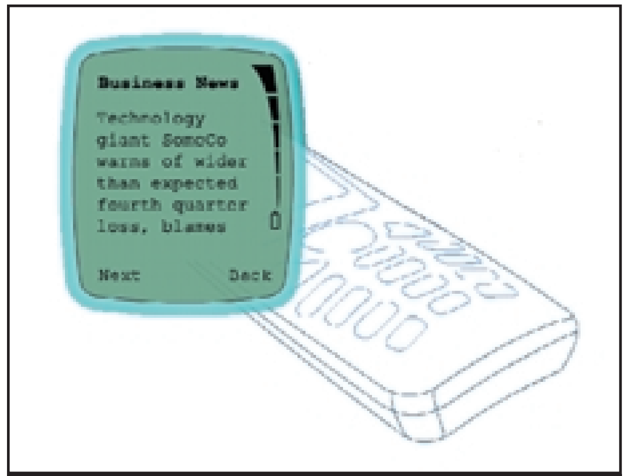

Business News
Technology giant SomeCo warns of wider than expected fourth quarter loss, blames
Next
Back

FIG. 5: A MOBILE PORTLET

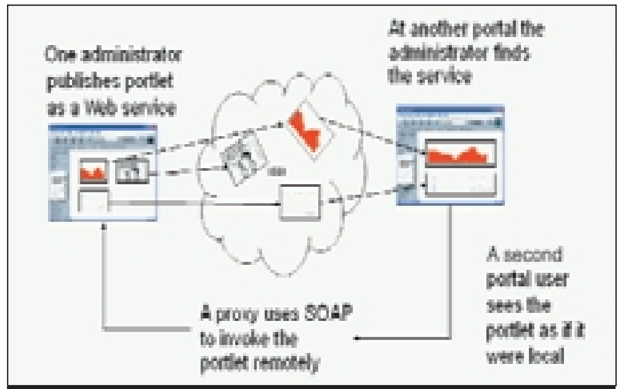

One administrator publishes portlet as a Web service
At another portal the administrator finds the service
A second portal user sees the portlet as if it were local
A proxy uses SOAP to invoke the portlet remotely

FIG. 6: REMOTE PORTLET WEB SERVICES

portlets to different users can enable those users to perform specific administration tasks. It's not necessary to grant all administration rights or to add those users to a special administrator's group.

Administration portlets have special access to the internal data structures and programming interfaces of WebSphere Portal. These portlets include functions such as installing new portlets, changing portlet configuration parameters, setting access rights, creating themes and skins, and defining page layouts and contents.

WebSphere Portal also provides a batch XML interface to export and import portal definitions. The XML interface can be used to back up and restore entire portal configurations or just to copy parts of the configuration, such as certain pages or page groups and portlets.

Collaboration

In WebSphere Portal Extend, collaboration functions are introduced. The idea is that instead of thinking about going to a separate product to collaborate with your team members, you can add interesting collaboration functions to any portlet. Collaborative Components are APIs and JSP tag libraries for integrating the functionality of Lotus Domino, Lotus Sametime, Lotus QuickPlace, and Lotus Discovery Server into the portal. Instead of working with more complex product APIs, collaborative components provide developers with an easier way to integrate Lotus applications into the portal.

- ***People Services:*** Determines who is online and what their status is.
- ***Domino Data Service:*** Used to create and edit documents in Domino databases, and to work with views.
- ***Discovery Services***: For finding people, documents, knowledge maps, and profiles.
- ***QuickPlace Service***: Creates new Lotus QuickPlace team rooms.
- ***Menu Service***: Displays a popup menu in the portal.

Remote Portlet Web Services

Remote portlet Web services are special Web services designed to plug in to portals. They conform to a well-defined Web services interface description with a fixed set of Web methods that follow the Portlet API. Since the interface is fixed, remote portlet Web services can be invoked through a generic proxy (see Figure 6).

Portlets created for WebSphere Portal are automatically enabled as remote portlet Web services. However, remote portlet Web services may be hosted on other systems and be implemented in programming languages other than Java, as long as the implementation follows the WSDL contract that is defined by the portal server.

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

When the portal receives a SOAP request from a remote portlet, it analyzes contextual information of the remote request and sets up a local context including user and client device–type information. After setting up the context it invokes a local proxy for the remote portlet. Portlet proxies are local portlets that don't implement any functionality themselves but rather act on behalf of remote portlets. Whenever the portlet container invokes them, they invoke the remote portlet running on the remote server and pass the results back to the local container.

Summary

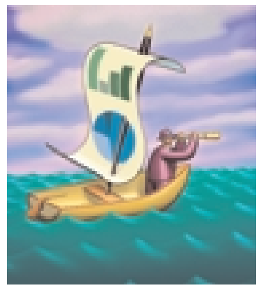
WebSphere Portal Server is designed to allow anyone to create and publish Web content or Web applications so they can easily and flexibly be composed into pages. This helps companies unify and reuse existing applications, giving them a longer useful life. It improves operational efficiency and security, and gives greater user interface consistency. Ultimately, the goal is increased employee productivity, better collaboration, and quicker access to more relevant information.

Chart Your Course to Success in IT...

...order your copy of Java Trends: 2003

Available July 1*

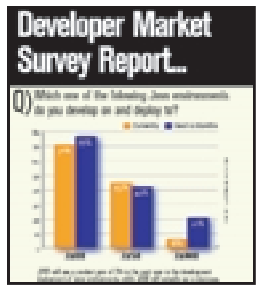
Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?



EnginData Research has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.

EnginData's research is invaluable to leading IT vendors, integrators, Fortune 500 companies, trade-show organizers, publishers, and e-business organizations worldwide.



Here's just a sample of the critical data points the Java report delivers...

- ✔ Current and projected usage of languages and platforms
- ✔ Multiple rankings of hundreds of software vendors and tools
- ✔ Types of applications being developed
- ✔ Databases being used
- ✔ Purchasing patterns
- ✔ Company size
- ✔ Development and purchasing timelines
- ✔ Perceptions and usage of Web services
- ✔ Preferred Java IDE
- ✔ J2EE, J2ME, J2SE usage comparisons

As an IT specialist, **EnginData Research** focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- ✔ Time frame for developing Web services – enabled apps
- ✔ Percentage of apps with Web services today
- ✔ Sourcing and hosting Web services
- ✔ Perceived leaders in Web services tools and solutions

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of **EnginData Research**. Order your copy of the 2002–2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.

engindata RESEARCH

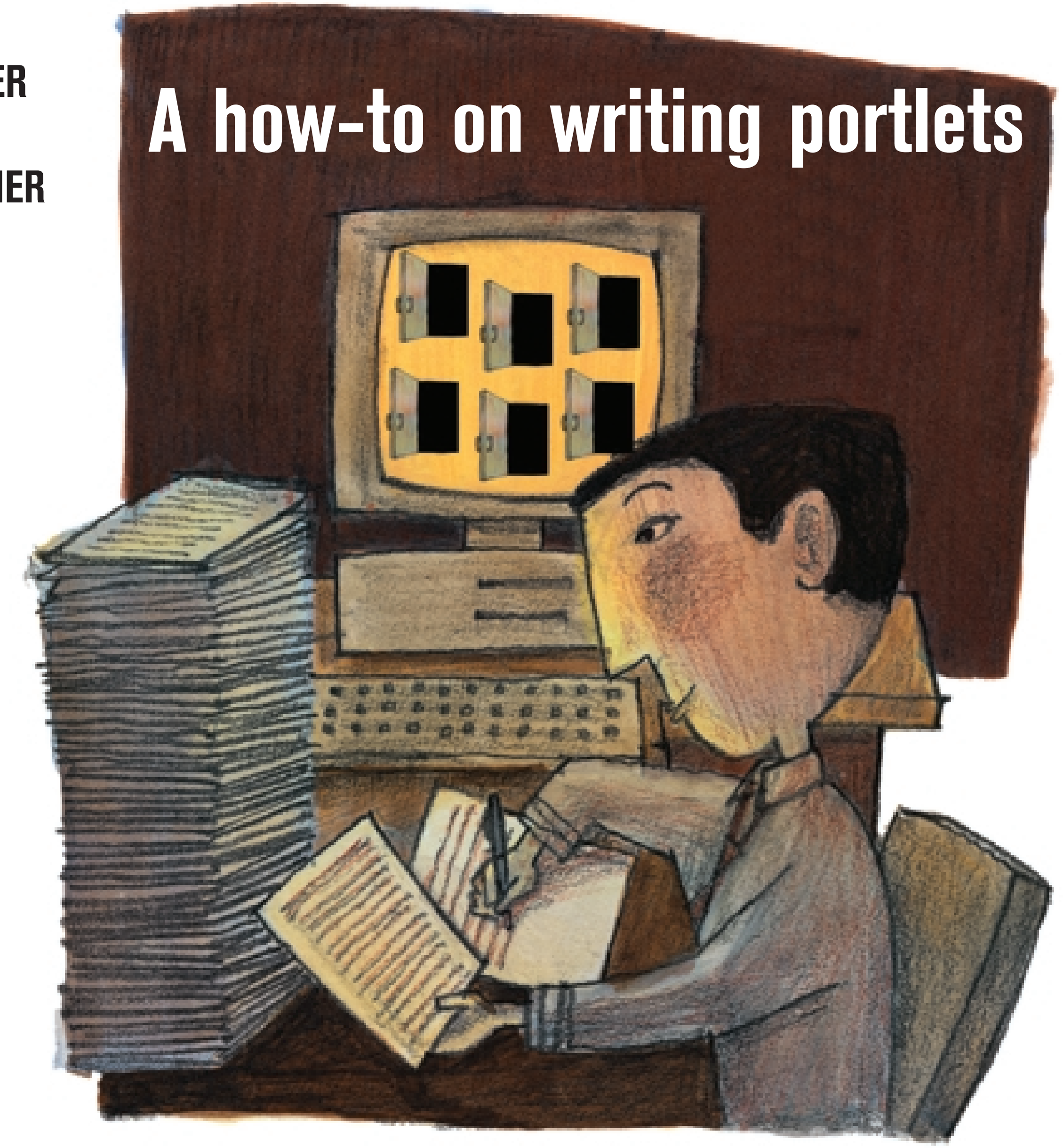
www.engindata.com

*A limited number of preview copies will be available at our booth (#624) at JDJEdge International Java Developer's Conference & Expo, June 24–27, at the Jacob Javits Convention Center in New York.

ENTERING THE PORTLET ERA

A how-to on writing portlets

BY
PETER FISCHER
AND
STEPHAN HESMER



Portals are Web sites that serve as a jumping-off point to information and applications on the Internet or from an intranet. To accommodate the aggregation and display of diverse content in a dynamic manner, a portal server must provide a framework that breaks the different portal components into pluggable portlets.

ABOUT THE AUTHOR
Peter Fischer is an IT specialist and software developer for IBM WebSphere Portal.
E-MAIL
peter.fischer@de.ibm.com

ABOUT THE AUTHOR
Stephan Hesmer is an IT specialist and software developer for IBM WebSphere Portal.
E-MAIL
stephan.hesmer@de.ibm.com

Each portlet is responsible for providing specific information and must therefore perform tasks like accessing content from its source (for example, a Web site, database, or e-mail server) and transforming the content so it can be rendered to the client most efficiently. The IBM WebSphere Portal for Multiplatforms Version 4.1 provides a framework of services to make the task of writing and managing portlets as easy as possible.

From a user's perspective, a portlet is a small window in the portal page that provides a specific service or information, for example, a calendar or news feed portlet. From an application development perspective, portlets are pluggable modules designed to run inside the portlet container of IBM WebSphere Portal 4.1.

Portlets are able to use the portal infrastructure to access user profile information, participate in window and action events, communicate with other portlets, access remote content, look up credentials, and store persistent data. To make performing these tasks very comfortable, the Portlet API provides standard interfaces for these functions.

Portlets and Servlets

All portlets are based on the abstract Portlet class, which is the central abstraction of the Portlet API.

Portlets are a special type of servlet, with properties that allow them to easily plug into and run in the portal server, plus various other features. They also have some natural restrictions: unlike servlets, portlets cannot forward requests, write arbitrary markup to the output stream, or send redirects and errors directly to browsers.

Generally, portlets are administered more dynamically than servlets. The following updates can be applied without having to start and restart the portal server:

- Portlet applications consisting of several portlets can be installed and removed dynamically using the portal administration user interface.
- The settings of a portlet can be changed on the fly by an administrator.

The portlet container relies on the J2EE architecture implemented by IBM WebSphere Application Server. Portlet developers benefit from this in a variety of ways:

- The Portlet API leverages the well-known Servlet API, extending it where necessary.
- Developers familiar with the Servlet API are able to instantly write portlets.
- Developers can use existing tools to develop portlets and JSP.
- Because portlets are part of the J2EE model, developers have a variety of options to combine J2EE-compliant technologies.
- Developers can reuse simple existing servlets/JSP with little or no effort.

Portlets are packaged in WAR files similar to J2EE Web applications and are deployed like servlets. Like other servlets, portlets are defined to the application server using the Web application deployment descriptor (web.xml). This file defines the portlet's class file, the servlet mapping, and read-only initialization parameters.

In addition to the servlet descriptor, portlets must also provide a portlet deployment descriptor (portlet.xml) to define the portlet's capabilities for the portal server. This information includes configuration parameters specific to a particular portlet or portlet application, as well as general information that all portlets provide, such as what type of markup the portlet supports. The portal server uses this information to provide services for the portlet. For example, if a portlet registers its support for help and edit modes in the portlet deployment descriptor, the portal server will render icons to allow the user to invoke the portlet's help and edit pages.

Portlet Applications

Portlet applications provide the means to package a group of related portlets that share the same context. The context contains all resources, e.g., images, properties files, and classes. All portlets must be packaged as part of a portlet application.

Portlet applications provide no code on their own but form a logical group of portlets. Besides this more logical gain, portlets of the same portlet application share their session and can also exchange messages.

Let's Start

Before you can start developing a portlet, you need to set up a portlet development environment – and a runtime environment wouldn't be bad idea either. This environment can be set up on one machine, but for performance reasons, we suggest using two machines.

THE DEVELOPMENT ENVIRONMENT

- ***JDK:*** WPS was compiled using JDK 1.3.0 (the JDK release shipped with IBM WebSphere Application Server 4.0.2).
- ***Editor:*** You'll also want to install your favorite source-code and HTML editors.
- ***J2EE Tools:*** Some example portlets in this article use JSP components to output markup to the user's client machine. IBM WebSphere Studio Application Developer is a good tool for editing these.
- ***Libraries:*** The following Java archive (JAR) files are required: portlet-api.jar, wps.jar, wpsportlets.jar, j2ee.jar, and websphere.jar.

THE RUNTIME ENVIRONMENT

On this machine you need to install IBM WebSphere Application Server 4.0.2, along with the latest set of patches. If you want this machine to have a fully functional version of WPS, you also need to install IBM DB2 Universal Database Version 7.2. These are prerequisite products for the developer release of WPS.

Once you've installed and tested the software on both machines, you need to add references to the following JAR files, required for portlet development, to your development machine's CLASSPATH:

- was_root/lib/app/portlet-api.jar
- was_root/lib/app/wps.jar
- was_root/lib/app/wpsportlets.jar
- was_root/lib/j2ee.jar
- was_root/lib/websphere.jar

Note that was_root is the directory where IBM WebSphere Application Server is installed.

Now that you have a complete development environment, it's time to do some portlet programming!

The obligatory first portlet: HelloWorld! Listing 1 shows a simple portlet structure. HelloWorld, while not highly functional, shows the basic structure of a portlet. A simple portlet requires a single method, the service method. A portlet's service method is called when the portal is requesting that the portlet render its data. In this sample this code snippet contains the data:

```
<p>Hello Portal! This is my first port-
let!</p>
```

Portlets packaged together in a single WAR file may share resources like images, JSP components, stylesheets, and other resources.

You need to create a Web deployment descriptor, web.xml (see Listing 2), and a portlet deployment descriptor, portlet.xml (see Listing 3), and include these XML documents in your HelloWorld WAR file so you can package and run the HelloWorld sample.

The portlet deployment descriptor describes the portlet application, each portlet in the application, and each portlet's title and capabilities. For a complete description of the portlet deployment descriptor, consult the IBM Portlet Development Guide (see Resources).

PACKAGING THE PORTLET APPLICATION

Now that you have the source code for the portlet and the portlet descriptor, you can package the pieces into a WAR file. Lay out the source code and portlet descriptor document in a directory structure such as the one shown below, which will make the building of the portlet and WAR file an easy process.

```
helloworld.war
__META-INF
_   MANIFEST.MF
__WEB-INF
  _ portlet.xml
  _ web.xml

  _
  __classes
    __com
      __ibm
        __wps
          __samples
            __helloworld
                HelloWorld.java
```

Using this directory structure you can create a batch file that compiles the portlet and builds the associated WAR file. This batch file assumes the JDK CLASSPATH has been set up to point to the portal JAR files required to compile this portlet. For this example we named this batch file BuildEx1.bat and placed it in the HelloWorld directory. Be sure to modify line 2 of the batch file to point to the base directory where you stored your portlet files, as shown in the following code snippet. Note that some lines (here and in the following examples) have been split so they'll display.

```
@echo off
set bd=D:/HelloWorld
javac -d
  %bd%/WEB-INF/classes
  %bd%/WEB-
INF/classes/com/ibm/wps/samples/helloworld/He
lloWorld.java
jar -cf HelloWorld.war WEB-INF
```

Running this batch file compiles the portlet and builds the WAR file. The next step is to run the portlet application on your runtime portal machine via the Install Portlet, as described in the WebSphere InfoCenter (see Resources).

Stocks Portlet Sample

To show some of the concepts and techniques associated with portlet development, we'll build a simple Stocks sample portlet, explaining each piece of the API we come across.

In the VIEW mode the portlet shows the stock values, which can be configured in the EDIT mode.

Portlet modes allow a portlet to display a different user interface, depending on the task required of the portlet. The following modes are provided by the Portlet API:

- *VIEW:* When a portlet is initially constructed on the portal page for a user, it is displayed in its VIEW mode. This is the portlet's normal mode of operation.
- *HELP:* If this mode is supported by a portlet, the portlet provides a help page for users to obtain more information about it.
- *EDIT:* If this mode is supported by a portlet, it provides a page for users to customize the portlet for their own needs. For example, a portlet can provide a page for users to specify a location for obtaining local weather and events.
- *CONFIGURE:* If this mode is supported by a portlet, it provides a page for portal administrators to configure the portlet for a user or a group of users.

The Portlet API provides methods for the portlet to determine the current mode: One way would be to call PortletRequest.getMode() in the service() method. An easier way would be to extend the PortletAdapter class, which provides methods for each mode, e.g., doView(), which is called if the portlet should be rendered in the VIEW mode.

This is similar to HttpServlet, which provides methods such as doPost(), in addition to the service() method.

```
…
public void doView (PortletRequest request,
                    PortletResponse response)
    throws PortletException, IOException
{
…
}
…
```

What functionality do we provide in this example? In the VIEW mode, the portlet lists all user-defined stock symbols with their current values. To keep it simple, the current values will just be random numbers. To add some interaction, it's possible to hide or show a description for the stock symbols on request (by clicking a link). In the EDIT mode you're able to add new stock symbols by entering a name and a description. The user-defined stock symbols are listed here as well but without their value or description. Additionally, it's possible to delete the stock symbols. To keep it simple, we only provide the option to delete all stock symbols.

How do we implement and prepare the portlet's basic functionality? First, we want to use JSP to render the content. This has the big advantage of automatically using the Model-View-Controller concept with the configured stock symbols as model, the JSP as view, and the portlet code as controller.

After the model and the controller (which are in the portlet code) are implemented, we can dynamically change the view, enabling us not only to easily work on the layout but also making it easy to debug, as the JSP can access the Portlet API as well. So we can put debugging statements into the JSP without having to redeploy or update the portlet itself.

In our example portlet we configure the name of the JSP within the web.xml (see Listing 4). These names are accessible via the PortletConfig:

```
…
String jspName =
getPortletConfig().getInitParameter(VIEW_JSP);
…
```

The PortletConfig file provides the Portlet class with its initial configuration. This information is valid for every concrete portlet derived from the portlet.

A portlet's configuration is set by the portlet developer and initially read from its associated servlet from the Web deployment descriptor. The configuration is read-only and cannot be changed by the portlet. The PortletConfig is passed to the portlet in the init() method of the abstract Portlet class and is used to access portlet-specific configuration parameters using getInitParameters().

After getting the name, the JSP is included by using the PortletContext:

```
…
getPortletConfig().getContext().include("/WEB-
INF/jsp/"+jspName, request, response);
…
```

The PortletContext interface defines a portlet's view of the portlet container within which each portlet is running. The PortletContext also allows a portlet to access resources available to it. Using the context, a portlet can access the portlet log, access context parameters common to all portlets within the portlet application, obtain URL references to resources, or access portlet services.

Now that we're able to set configuration parameters, access them in the source code, and call/include resources, let's start implementing the portlet.

First we list the stock symbols in the view JSP previously configured in the EDIT mode and stored persistently in PortletData.

The PortletData is a simple means for the portlet to store data persistently on the instance level, allowing each portlet instance to store and read its own data. The portlet may get, set, and remove attributes during one request. To commit the changes, the store() method has to be called, otherwise all data is lost after the request. The data is read-only and can be written by the portlet only when it is in EDIT mode.

In our example we just want to read them in the VIEW: get all the symbols and loop through the parameters, displaying their name (attribute name), their description (attribute value) if required, and a random number as stock value (see Listing 5).

In addition, we want a link to switch to symbol details. Such a link is to be implemented on the base of a PortletURI object.

The PortletURI object contains a URI pointing to the portlet instance and can be further extended by adding portlet-specific parameters and by attaching actions (see EDIT mode below). The complete URI can be converted to a string that's ready for embedding into markup. The PortletResponse provides us with all necessary methods to create such a URI.

The PortletResponse encapsulates information to be returned from the server to the client by using a Java

PERFORMANT

WWW.PERFORMANT.COM

PrintWriter and is passed via the beginPage(), endPage(), and service() method. The response also includes methods for creating the PortletURI object (as mentioned above) or qualifying portlet markup with the portlet's namespace.

Each portlet runs in its own unique namespace. The method encodeNamespace() is used by portlets to bring attributes into the portlet's output to avoid name clashes with other portlets on the same page. Attributes can include parameter names, global variables, or JavaScript function names.

After creating the URI we can add parameters or actions to it and convert it to a string that can be used within our HTML link.

As the link is put together in the JSP, the URI is given to the JSP in a bean that also contains the message for the link plus a Boolean value to indicate whether the description will be shown (see Listing 6). We used that value above when listing the stock symbols. This bean is added to the request, from which the JSP can get it again via the JSP useBean tag.

In the EDIT mode we want to enable users to personalize their portlet with some stock symbols. These symbols are stored in the PortletData as shown below:

```
 …
            PortletData data =
event.getRequest().getData();

            data.setAttribute(name, desc);

            …

            data.store();
…
```

The data to be stored is collected via the StocksEdit.jsp's form. Submitting the form triggers a PortletAction.

Actions are portlet-specific activities that need to be performed as result of the incoming request, but before the service() method of the portlet is called. For example, when a user is entering the data in the portlet's edit mode and clicks the "Save" button, the portlet needs to process the posted data before the next markup is generated. This is the case, as the action brings the new stock symbol to the portlet, which then wants to display the new symbol right away. This can be achieved by adding a "Save" action to the URI that represents the "Save" button (see Listing 7).

For the portlet to receive the action it must implement the ActionListener interface as shown:

```
…
// ActionListener interface method
    public void actionPerformed (ActionEvent
event) throws PortletException
    {

        DefaultPortletAction action =
(DefaultPortletAction) event.getAction();

        …
    }
```

The content of the form can be retrieved as a parameter of the PortletRequest to be stored as shown:

```
if
(action.getName().equalsIgnoreCase(STOCKS_ADD
))
{
```

LISTING 1

```
package com.ibm.wps.samples.helloworld;

import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlets.*;
import java.io.*;

public class HelloWorld extends PortletAdapter
{
  public void service( PortletRequest portletRequest,
                       PortletResponse portletResponse)
  throws PortletException, IOException
  {
    writer = portletResponse.getWriter();
    writer.println("<p>Hello Portal! This is my first Portlet!</p>");
  }
}
```

LISTING 2

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app id="WebApp_1">
  <display-name>HelloWorld</display-name>
  <servlet id="Servlet_1">
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>com.ibm.wps.samples.helloworld.HelloWorld</servlet-
      class>
  </servlet>
  <servlet-mapping id="ServletMapping_1">
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld/*</url-pattern>
  </servlet-mapping>
</web-app>
```

LISTING 3

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application
1.1//EN" "portlet_1.1.dtd">
<portlet-app-def>
  <portlet-app uid="com.myCompany.myPortletApp.54321">
    <portlet-app-name>HelloWorld Portlet Application</portlet-app-name>
    <portlet id="Portlet_1" href="WEB-INF/web.xml#Servlet_1">
      <portlet-name>My portlet</portlet-name>
      <allows>
        <minimized>
        <maximized>
      </allows>
      <supports>
        <markup name="html">
          <view output="fragment"/>
        </markup>
      </supports>
    </portlet>
  </portlet-app>
  <concrete-portlet-app uid="com.myCompany.myConcretePortletApp.54321">
    <portlet-app-name> HelloWorld Portlet Application - Example Portlet
      #1</portlet-app-name>
    <concrete-portlet href="#Portlet_1">
      <portlet-name>My concrete portlet</portlet-name>
      <default-locale>en</default-locale>
      <language locale="en_US">
        <title> HelloWorld - Example Portlet #1</title>
        <description>HelloWorld - Example Portlet #1</description>
      </language>
    </concrete-portlet>
  </concrete-portlet-app>
</portlet-app-def>
```

LISTING 4

```
…
      <servlet id="Servlet_1">
         <servlet-name>Stocks</servlet-name>
         <servlet-class>com.mycompany.portlets.stocks.Stocks
           Portlet</servlet-class>
         …
         <init-param>
            <param-name>jsp.view</param-name>
            <param-value>stocksView.jsp</param-value>
         </init-param>
         <init-param>
            <param-name>jsp.edit</param-name>
            <param-value>stocksEdit.jsp</param-value>
         </init-param>
      </servlet>
…
```

```
  String name = (String)
event.getRequest().getParameter(STOCKS_NAME);
  String desc = (String)
event.getRequest().getParameter(STOCKS_DESC);
  …
}
```

The PortletRequest object is providing the portlet with request-specific data and the opportunity to access information such as:

- ***Attributes:*** Attributes are name/value pairs associated with a request. Attributes are available only for the scope of the request. The portlet can get, set, and remove attributes during one request.
- ***Parameters:*** Parameters are name/value pairs sent by the client in the URI query string as part of a request. Often the parameters are posted from a form. Parameters are available for the scope of a specific request. The portlet can get – but not set – parameters from a request.

Now we have the code for entering new stock symbols and we have seen how to display them in the VIEW mode; that leaves us with the function to delete the symbols. This is just another action that's put into a link within the edit JSP:

```
…
<A
HREF='<portletAPI:createURI><portletAPI:URIAct
ion
name='<%=StocksPortlet.STOCKS_DELETE%>'/></por
tletAPI:createURI>'>delete all</a>
…
```

With help of the actions' names we can identify what to do in the ActionListener (see Listing 8)

Conclusion

This article only scratches the surface of the potential in portlet programming. We've shown how to get started in portlet programming, including setting up a portlet development and runtime environment. We used the HelloWorld portlet to demonstrate a very basic portlet and give you a feel for the basic structure of a portlet, as well as how to build, package, and deploy a portlet on a portal server. We also introduced the concepts of portlet modes and portlet window states, the use of JSP components and JavaBeans to render data in a portlet, and portlet action handling with the help of the stocks portlet.

These concepts make up the basics of portlet programming. If you combine these concepts with your existing application's data connectors, you will be able to build powerful portlets and bring your product into the portal era of the Internet.

To get some more practice in developing portlets, we suggest extending the example portlet step by step with additional functions. As a starting point you could implement the deletion of single stock quotes.

Resources

- *WebSphere Portal Server:* www-4.ibm.com/software/webservers.
- *WebSphere InfoCenter:* www-4.ibm.com/software/webservers/portal/library/InfoCenter.
- *IBM Portlet Development Guide:* www-4.ibm.com/software/webservers/portal/library.

LISTING 5

```
<%
Enumeration stocks = portletRequest.getData().getAttributeNames();

while (stocks.hasMoreElements())
{
  String symbol = (String) stocks.nextElement();
%>
… <%=symbol%>  …
<%
  if (stocksBean.isShowDescription()) {
%>
…  <%=portletRequest.getData().getAttribute(symbol)%>  …
<%
  }
%>
…  <%=Math.round(Math.random() * 1000)%>.00 $  …
…
```

LISTING 6

```
        StocksBean bean = new StocksBean();

        boolean show = request.getParameter(STOCKS_DETAIL) != null;
        bean.setShowDescription(show);

        PortletURI uri = response.createURI();

        if (show)
        {
            bean.setMessage("Hide Description");

        }
        else
        {
            bean.setMessage("Show Description");
            uri.addParameter(STOCKS_DETAIL, "true");
        }

        bean.setUrl(uri.toString());

        request.setAttribute("stocksBean", bean);

…
<jsp:useBean id="stocksBean"
class="com.mycompany.portlets.stocks.StocksBean" scope="request" />
…
…
<A HREF='<%=stocksBean.getUrl()%>'><%=stocksBean.getMessage()%></A>
…
```

LISTING 7

```
<FORM method='POST' action="<portletAPI:createURI><portletAPI:URIAction
name=
      '<%=StocksPortlet.STOCKS_ADD%>'/></portletAPI:createURI>"  …
  Name: …        <INPUT name='<portletAPI:encodeNamespace
  value="<%=StocksPortlet.STOCKS_NAME%>"/>'>
  …
  Description: … <INPUT name='<portletAPI:encodeNamespace
  value="<%=StocksPortlet.STOCKS_DESC%>"/>'>
  …
  <INPUT type='submit' name='addButton' value='Add'>
  …
</FORM>
```

LISTING 8

```
    // ActionListener interface method
    public void actionPerformed (ActionEvent event) throws
    PortletException
    {

        DefaultPortletAction action = (DefaultPortletAction)
        event.getAction();

        if (action.getName().equalsIgnoreCase(STOCKS_ADD))
        {
           …
        }
        if (action.getName().equalsIgnoreCase(STOCKS_DELETE))
        {
            event.getRequest().getData().removeAllAttributes();
            …
            event.getRequest().getData().store();
            …
        }
    }
```

A new way to update from multiple sources

Web Content Publisher

BY ROB WILL

Content is core to many portal implementations. Content is often what brings visitors to a portal, and up-to-date and relevant content is what keeps them coming back. Some content is retrieved programmatically. For example, a bank account balance or airline reservation information is retrieved using an application. But a lot of the content within most portals is informational in nature – an overview of all the accounts that the bank offers, details on each bank account, information on investing in stock, or information on the latest security procedures at the local airport. The owners of the site create this type of content using various tools, often making use of some sort of Web content management solution.

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 on providing tooling that helps WebSphere customers create and target content for their Web sites.

WebSphere Portal's content can be managed using many of the leading Web content management solutions. WebSphere Portal and many of the leading Web content management solution providers have worked together to produce integration kits that make it fast and easy to integrate each solution with WebSphere Portal.

In addition, WebSphere Portal includes a Web Content Publisher (WCP) capability that provides some of the basic Web content development needs of some portal customers. WCP provides:

- Browser-based editing of template-based content and file content
- Authoring templates written in JSP or XSL for creating content
- Generation templates written in JSP or XSL for generating runtime view of the content in various markup languages
- Workflow and approval of content
- Support for syndicated content
- Versioning and access control
- Tight integration with
  - WebSphere Portal
  - WebSphere Personalization
  - WebSphere Studio Application Developer and Site Developer Tooling
  - WebSphere Edge Server
  - WebSphere Application Server

Creating and Editing Content

WCP provides a Web-based application for creating and editing Web content. As shown in Figure 1, the content publishing application provides a typical folder view on the left and a list of content on the right. On the bottom of the page are the properties of the selected content.

To edit file content, you simply select the content, click on the edit button, and your favorite editor for that type of content is launched. For example, when you edit a Word document Microsoft Word might be launched; when you edit an image, WCP might launch PerfectPhoto (see Figure 2).

While it's handy to be able to edit file content such as Word documents and images, the real power of WCP is its ability to allow business users to create Web content using template-driven or structured content. With structured content, the Web team develops input forms (called authoring templates) and JSPs or XSL files for generating views of the structured content. The user only enters the values within the form and then WCP generates the views of the content for the portal. The generated views might differ in terms of the amount of detail shown or markup language used to create the view. For example, if the business user created the information for a new toy, then WCP could generate a summary list of all the toys, including the newly added toy, and also a more detailed view the toy. This style (a summary list linked to a more detailed view) is a common paradigm for portlets. Figure 3 shows how the input form might look. The generated views might look something like Figure 4.

There are many advantages to developing content using templates and structured content. Template-driven content allows business users to contribute the content of a page or part of a page without knowing anything about formatting the page. This is especially powerful because the Web development team maintains control over the look and feel, style, and navigation of the site while the business users are empowered to contribute their knowledge to the portal and to control the actual content and messages. Another advantage of using structured content and templates is that a single content item can show up in multiple places within the portal, with different levels of detail (views), without the content having to be re-entered or maintaining multiple copies. If some of the views within the portal are personalized, the structured content can be used within rules to select summary views targeted to each visitor, as well as within static summary views, again without having to re-enter the data multiple times. Finally, each content item can be formatted so it can be shown to a variety of devices (HTML, wireless, etc.), without needing to re-enter the actual content.

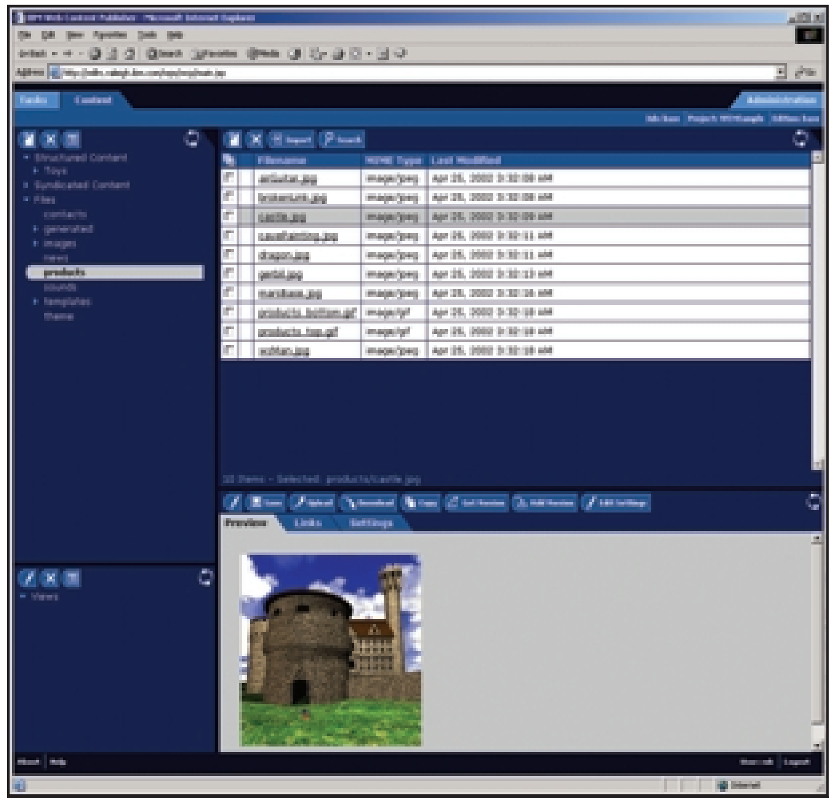


FIG. 1: CONTENT PUBLISHING APPLICATION

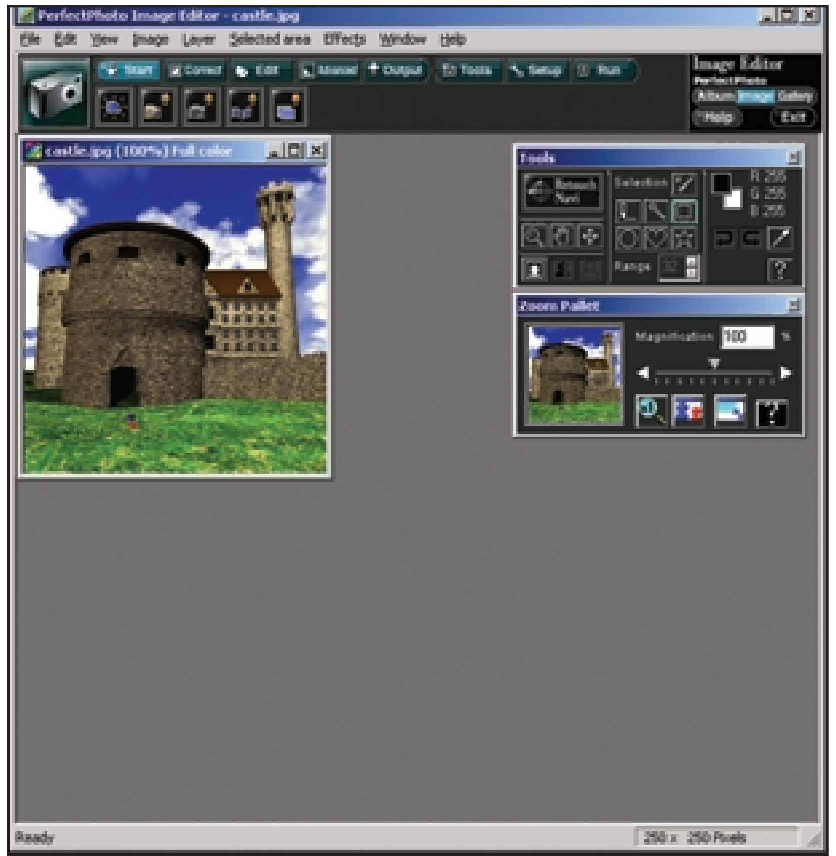


FIG. 2: CONTENT PUBLISHING APPLICATION

Templates

Members of the Web team typically define the authoring and generation templates. Four types of authoring templates are used by content contributors: add, edit, preview, and show. Add templates create a new instance of a given type of structured content. Edit templates change an existing instance of a given type of structured content. Add and edit templates are typically HTML forms and may look very much the same, the only difference being that the add template form will be prepopulated with default values while the edit template form will be prepopulated with the existing values for each field. Preview templates give the author an idea of how the structured content might look when formatted for the portal. They are typically one of the generation templates, although that isn't required. Show templates are a cross between edit templates and preview templates. Show templates show all of the properties of an existing instance of a given type of structured content, with an emphasis on showing the data, not how that item might look on the portal.

ADD AND EDIT TEMPLATES

Add and edit templates are HTML forms that post the same servlet within WCP. The form within the add and edit template should be defined with this line:

```
<FORM ACTION="/wps/wcp/com-
mand" METHOD="POST">
```

and must include two hidden fields – either

```
<INPUT TYPE="hidden"
NAME="command"
VALUE="addItem"/>
```

or

```
<INPUT TYPE="hidden"
NAME="command"
VALUE="updateItem"/>
```

and

```
<INPUT TYPE="hidden"
NAME="beanName"
VALUE="WCMSample.Product"/>
```

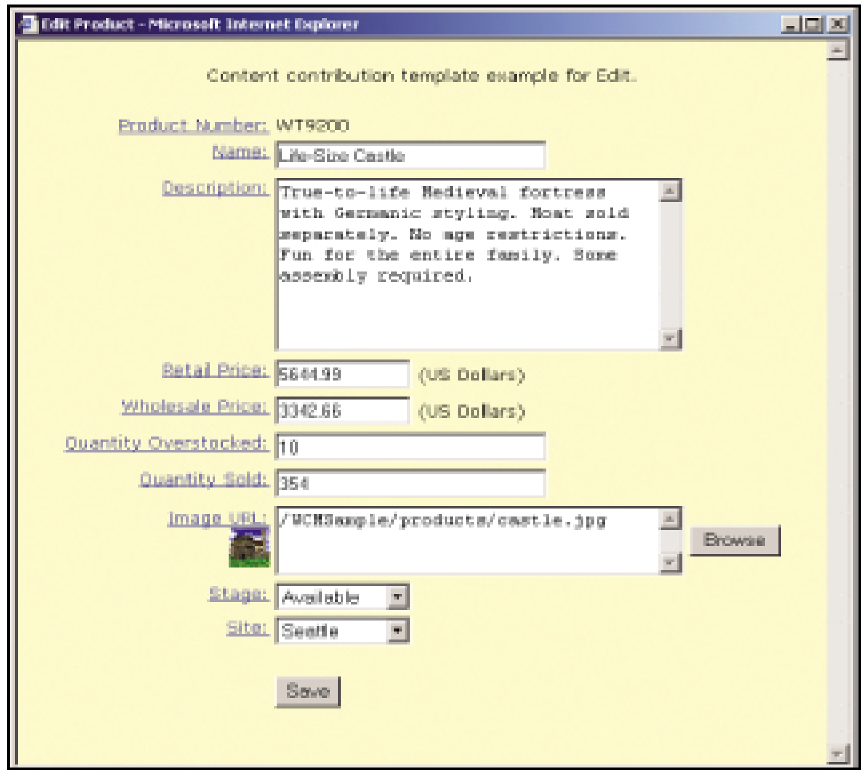
The command parameter tells the WCP servlet whether this is new content (addItem) or an update to an existing item (updateItem). The beanName parameter tells the WCP servlet the name of the resource class that represents this type of structured content.

The rest of the form contains input fields for the properties of the resource class that you are creating or updating. The name of each input field must match the corresponding property name of the resource class. You can use any type of form input field that is valid with your browser.

If you're creating an edit form, you'll want to prepopulate it with the current values of the item. When the business user hits the Edit button, WCP passes the resource instance to your edit form as an instance of the corresponding resource object. To define this object to your JSP edit page, you must include the two useBean tags shown in Listing 1.

The first useBean tag is the same for every edit JSP. The second useBean tag is the same for all edit JSPs, except that the type parameter should be changed to the resource type this form is for.

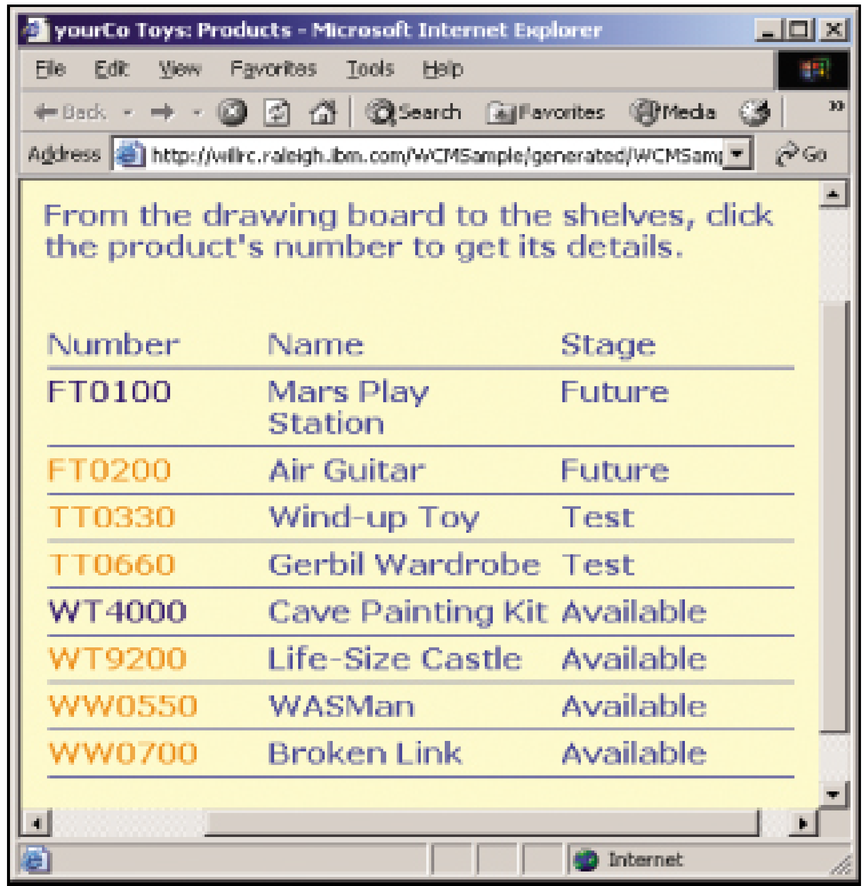
After defining these two beans, you can prepopulate the form using simple scriptlet logic. For example:



Content contribution template example for Edit.

Product Number:	WT9200
Name:	Life-Size Castle
Description:	True-to-life Medieval fortress with Germanic styling. Moat sold separately. No age restrictions. Fun for the entire family. Some assembly required.
Retail Price:	5644.99 (US Dollars)
Wholesale Price:	3342.66 (US Dollars)
Quantity Overstocked:	10
Quantity Sold:	354
Image URL:	/WCMSample/products/castle.jpg
Stage:	Available
Site:	Seattle

FIG. 3: INPUT FORM


yourCo Toys: Products - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://willrc.raleigh.ibm.com/WCMSample/generated/WCMSam Go
Internet

From the drawing board to the shelves, click the product's number to get its details.

Number	Name	Stage
FT0100	Mars Play Station	Future
FT0200	Air Guitar	Future
TT0330	Wind-up Toy	Test
TT0660	Gerbil Wardrobe	Test
WT4000	Cave Painting Kit	Available
WT9200	Life-Size Castle	Available
WW0550	WASMan	Available
WW0700	Broken Link	Available

FIG. 4: ONE POSSIBLE GENERATED VIEWS

```
<INPUT TYPE="text"
NAME="RETAILPRICE"
VALUE="<%=
resObj.getRETAILPRICE()
%>"/>
```

And of course, you can define text areas, pull down menus, list boxes and other input types, and include all of the standard options on each INPUT tag.

PREVIEW AND SHOW TEMPLATES

Preview and show templates are created in exactly the same way. The only difference is in the information shown and how it's formatted. Of course, what is shown and the formatting is entirely up to you, so in some cases the preview and show templates might even be the same file. Preview/show templates must include the same two useBean tags as are included for edit templates. These are needed within the edit template. WCP will pass the selected content item as a resource instance to the preview/show template.

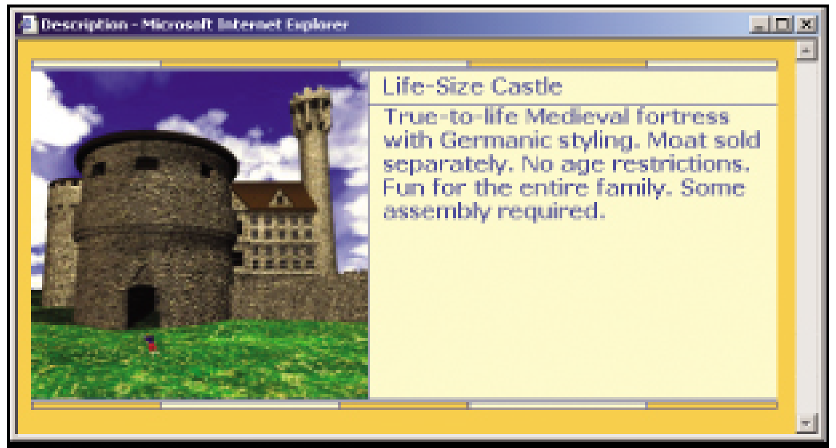
The preview/show template formats the instance of structured content using whatever HTML your browser supports. The property values are added to the HTML within the JSP using simple JSP expressions such as

```
<p>The amazing part is that
you get all of this for only
<%= resObj.getRETAILPRICE()
%>!!
```

GENERATION TEMPLATES

There are two types of generation templates. Summary templates create views that typically display a list of items allowing the user to select one of the items in order to get more detail. Detail templates create the more detailed views. As discussed earlier, this is a typical portlet paradigm.

Creating detail templates are exactly the same as creating preview templates (and you might want to use the same file for both preview and one of your detail views). There's no difference in how you specify the useBeans and display the content. The only reason that WCP has both a preview template and detail template concept is because you will often have multiple detail templates for each content type. You might want to show different views of the content within different parts of your portal, or you may want to generate views of each content item in HTML and other markup languages such as WML and VoiceXML. Since there can be multiple detail templates for any given content type, WCP needs to know which template to use when the author asks to preview a piece of structured content. There's no need to create a special preview template (unless you want to), but you do need to tell WCP which template to use for previewing.


Description - Microsoft Internet Explorer

Life-Size Castle

True-to-life Medieval fortress with Germanic styling. Moat sold separately. No age restrictions. Fun for the entire family. Some assembly required.

FIG. 5: ANOTHER POSSIBLE VIEW

Each time a piece of structured content is added or changed, WCP will regenerate all of the detail views for that piece of content automatically using the list of detail templates that you have defined. You can even use generation templates to generate resources from other resources. For example, you can generate a Press Release resource whenever a Product resource is changed. Such templates can be chained together, causing a cascade effect whenever a resource changes. In this way you can be sure the resources that relate to each other stay synchronized without manual effort.

Summary templates are a bit different from what we have discussed so far, but they have many similarities. The big difference is that instead of passing the template a single instance of structured content, WCP passes the template an array of resource instances. The template is then responsible for looping through the array and displaying the summary information desired, and typically creating links to the corresponding detail view. The useBean tag you include within a summary template is shown in Listing 2.

Within your JSP, you need to construct a loop to iterate through the items passed to your template. For example:

```
<%
    while(
contentList.hasMoreElements(
) )
    {
     WCMSample.Product prod-
uct =
(WCMSample.Product)contentLi
st.nextElement();
%>
```

Within the loop you display the desired information the same way you did in the other templates:

```
<%= resObj.getRETAILPRICE()
%>
```

Typically, one of the properties of each item would be a hyperlink to the detail view of that item (this is not required, and in some cases won't be done, since all you might want to do is show the summary list).

```
<A href="<%=
product.getPRODUCTNUMBER()
%>.html">
```

SIMPLEX KNOWLEDGE COMPANY

WWW.SKC.COM/TRAINING/PW420

In some cases, you'll construct the file name to reference differently. For example, it might be an extension other than HTML or the file name might have a qualifier pre-pended to it. The name for the generated detail view is specified when you define each detail template. WCP appends the resource ID to the file name that you specify to construct the file name under which the generated detail view will be saved as well as the directory name. The name of the file and directory for the generated summary view is also specified when you define the summary template.

Summary templates are assigned on a folder-by-folder basis and are generated on the same basis. To regenerate a summary view, select the corresponding folder and select Generate. WCP will pass off the structured content to your summary template for processing and save the results in the file name and location specified for that folder.

VIEWS

You can create views of your structured content by specifying search criteria based on the structured content's properties. While you can organize your structured content within folders, each item can be in only one folder. Views are similar to folders, but by using views you can further organize your content based on the criteria you select, and items can appear in more than one view. This is a handy way to organize your content across multiple dimensions.

You can assign summary templates to views in the same way you assign summary templates to other folders. This allows you to create summary portlets that provide different views for different purposes.

TEMPLATE WIZARDS

WCP's define template dialog has a wizard button that will generate each template type for you. While the formatting is not likely to be what you want, the useBean tags, loops, form fields, and property displays will all be correct. This makes it easy to build author and generation templates for your structured content. In addition, this wizard will surface your favorite editor to allow you to make changes to the generated template, preview the change, and continue this iterative development until you get the template to look exactly the way you want.

OTHER MARKUP LANGUAGES

Generation templates can generate summary and detail views in whatever markup language you need.

XSL TEMPLATES

In addition to JSP, you can use XSL to create any of the author or generation templates we've described. You can mix and match JSP and XSL templates within the same content type so that you can use the language that's best suited for each template type and markup language. We won't go into a lot of detail here on XSL templates, but they're very similar to JSP templates in terms of their purpose and how the forms are constructed. The major difference is that instead of passing resource object instances to the templates, WCP generates XML instances that represent the single content item or list of items and then invokes the XSL processor using the generate XML instance and your XSL template. XSL templates are particularly useful if you're using generation templates to generate JSPs since you cannot use JSP templates to generate JSP output. Easy-to-follow samples are included with WebSphere Portal.

PORTAL CONTENT ORGANIZER AND OTHER VIEWS

In addition to the static summary views that WCP generates, you can publish the actual resource objects to the production server. The content is stored in whatever runtime database you've chosen for your runtime repository, a database for example. In this case, you can create simple JSP portlets with embedded personalization rules in order to produce summary views personalized for each visitor.

You can also publish WCP-managed content to the Portal Content Organizer (PCO). PCO is a set of portlets that allow you to create private and shared customized views of content. PCO enables you to organize content into folders, as well as searching the content published to PCO.

Access Control

WCP allows you to assign privileges to content at the project, content type, folder, file extension (for example, .doc or .jpg), or individual item level. Privileges can be assigned to groups or individual users. Supported privileges are view, create, update, delete, and check links. In addition, there are system-level privileges, such as authorization to publish the project or to manage users.

The user identity is the authenticated uid from the user's LDAP directory entry. The LDAP directory is the LDAP server used to authenticate the users on the WebSphere Application server where WCP is installed. Group membership is determined from the group memberships of the authenticated user within this LDAP server.

Since WCP is used in developing the site, and not on the production site, it can be installed on a WebSphere Server with or without the Portal. If WebSphere Portal is used within your enterprise, then you can configure WebSphere Portal and WCP for single sign-on and link to the WCP application from the QuickLinks portlet. WCP requires that Domino be used as the authentication server, so when WCP and Portal are installed on the same WebSphere Server, Domino must be use for Portal authentication as well.

Workflow

WCP uses the Lotus Workflow (LWF) engine. WCP provides four sample workflow processes to get you started, but you can easily modify these or create your own using the Lotus Workflow Architect. Workflow Architect is a graphical process creation tool that enables you to easily define simple or complex processes that WCP will use for developing and approving content for your Web site.

Users making changes to content within a workflow job are automatically placed in a workspace unique to that job. Thus, the changes are scoped to the job. Users working on that job or approving the work can all preview the site as it will appear when the changes are published, while users not working on that job will continue to see the site according to the job they're working on. This provides the isolation needed so many users can contribute content at the same time without interference from other content contributors. After the changes are approved, they're promoted into the base edition of the site and become available for everyone to see. If the changes are rejected, they can be backed out without impacting the site, or they can be fixed iteratively until they are correct.

When users work on an activity within a job, they can share that activity with other users (so that they can make changes in parallel while seeing each other's changes), or assign the activity to another user so that that user can continue the updates that the first user was making. In this way, ad hoc parallel workflows and new steps can be added easily.

There is also a 'quick edit' mode that allows users to make changes that don't require workflow-based approval. For example, if the site required an emergency fix, a content contributor could enter 'quick edit' mode to make the change.

Syndicated Content

WCP supports the scheduled import of syndicated content over RSS channels. RSS is a popular syntax for expressing things like news stories in XML form. An RSS channel is just a collection of related stories contained in a single XML file. You can define as many channels as you like and WCP will retrieve the updates on each channel on a fixed schedule that you control. Syndicated content can be reviewed and accepted or rejected for inclusion on your site. WCP also provides a Java interface you can implement to categorize incoming channel content. You can define JSP or XSL transformations to convert the syndicated content data to any structured content model you like. There are a number of RSS providers – just search the Internet for 'rss' and you'll find them.

In addition, you can define summary templates in order to publish your own RSS channels. This allows you to use WCP to publish, not just consume, syndicated content. All you need to do is create a JSP or XSL template that generates an XML file with the appropriate channel format, with links to the detail views that you want the channel clients to be able to access.

WebSphere Studio Application Developer and Versioning

WCP provides workspace support that allows work-in-progress versions to be maintained separately from the base site until those changes are approved. However, in many cases you'll want to maintain additional versions of approved content in case you need to return to a previous version. WCP can be configured to use CVS to maintain versions of approved content. When CVS is used, the WCP user interface includes menu items that allow users to retrieve previous versions of any piece of versioned content or to create new versions.

When CVS is used, the WCP assets are stored within CVS as a WSAD project. WSAD developers can view and change the same content that WCP users work with. File content can be edited using the WSAD editors, while structured content is stored as XML files within the WSAD project and can be viewed and edited using the WSAD XML editor.

Editions

In addition to versioning individual assets, WCP can archive a version of the entire project. Project versions are called editions. Editions can be archived and then restored later. Editions also form the baseline for projects and changes. For example, after a project is published, if you create an edition out of that version of the project, then you can create two update streams off of that baseline. One update stream might be for fixes to the current Web site, while the other update stream might be for the long-term changes to create the next version of the site.

Edge of Network Support

WCP allows you to designate which pieces of content are 'cacheable.' When you publish content, WCP uses this property to decide which items are to be distributed using Edge's Content Distribution Framework. This allows you to take advantage of this powerful distribution facility to fan content out to a number of Edge servers.

Summary

Managing your portal content using a Web content management solution enables people throughout your company to contribute information and knowledge to your portal using template-driven content. Also, the information is more easily changed and as a result is more up-to-date and relevant to your portal's visitors. Content management solutions such as WebSphere Portal's Web Content Publisher component and our partner's Web content management solutions also allow you to tag content for personalization and other targeting strategies.

LISTING 1

```
<jsp:useBean class="com.ibm.wcm.GetGenericResource" id="getResource" type="com.ibm.wcm.GetGenericResource">
<% getResource.getResource(request); %>
</jsp:useBean>
<jsp:useBean id="resObj" type="WCMSample.Product" scope="request"/>
```

LISTING 2

```
<jsp:useBean class="com.ibm.wcm.GetGenericResourceList" id="getResourceList" type="com.ibm.wcm.GetGenericResourceList">
<% getResourceList.setRequest(request); %>
</jsp:useBean>
<% java.util.Enumeration contentList = getResourceList.getResourceList();
 if( contentList==null ) { redirect to another JSP… or take some other action  }
%>
```

The horizontal portal market in mid-2002

The Winds of Change

BY GENE **PHIFER** AND DAVID **GOOTZIT**

ABOUT THE AUTHOR
Gene Phifer is a VP and research director in Gartner's Internet Strategies Practice.

E-MAIL
gene.phifer@gartner.com

ABOUT THE AUTHOR
David Gootzit is a senior research analyst in Gartner's Internet Strategies Practice.

E-MAIL
david.gootzit@gartner.com

The portal product market has matured significantly since its birth in 1998. In the early days, 'pure-play' vendors provided the only options for enterprises evaluating portal products. This situation quickly changed as existing software vendors came to recognize the market desire for the functionality that the portal promised. Many of these companies, however, either misjudged the resources that would be required to thrive in this technology space or found themselves going down a path which few would-be customers were willing to follow. In either case, the result has been the same – while demand has increased for horizontal portal products, the market has refused to reward vendors lacking solid functionality or a strong vision. Pure-play portal vendors have been the primary casualties to this point. However, consolidation in the horizontal portal product market continues, and is likely entering its most critical stage. 2002 will bring major consolidation among the portal product vendors and some key, new lessons about portals.

Generally, vendors in the horizontal portal market are members of one of three groups – "pure-plays"; middle-tier, independent software vendors; or the large, independent software vendors, the "800 lb gorillas" of the technology world. For the most part, the pure-plays are pre-initial offering, with annual revenue of less than $40 million, fewer than 300 personnel, and under 200 customers, but they are focused on portals. Typically, a technology pure-play is not strong enough to field multiple complex products, although there are exceptions. The pure-plays were the pioneers of the portal market, and several continue to have significant mind share.

The middle-tier independent software vendors are mostly publicly held, have existed for longer than five years, and have revenues in the hundreds of millions of dollars from products that predate portals. Several of these vendors' interest in the portal market should be viewed as purely opportunistic, as they provided many of the cases of legacy applications with a Web front end suddenly being touted as "enterprise portals." However, this segment also contains vendors from backgrounds like middleware that have provided credible products.

The 800 lb gorillas are vendors with multibillion-dollar annual revenue, an established presence in multiple markets, and diverse product lines spanning systems, applications, consulting services, and software infrastructure. In many cases, these vendors entered the horizontal portal product marketplace later than those from the other two segments, but there is evidence that it's becoming their jungle.

Starting in the second half of 1999, the large, independent software vendors began to enter this market, but it's only in the last nine months that the market has felt their full weight. Gartner has seen a major power and momentum shift in the portal product market during this time. In the summer of 2001, the pure-play vendors held most of the market share and momentum. Since then, however, the balance of power has shifted to the 800 lb gorillas. The first round of consolidation of the portal product market was driven by the overcrowded nature of a market populated predominantly by small, privately owned, venture-capital-funded, unprofitable companies. One somewhat surprising trend within this shakeout has been the method. Rather than closing their doors, most of the pure-play vendors affected by the first round of consolidation were acquired. Examples of pure-plays that have been acquired since the beginning of 1991 include Sagemaker, Sequoia, DataChannel, TopTier, KnowledgeTrack, and Octopus.

This first round of consolidation is concluding with the departure of several middle-tier independent software vendors from the portal product market. Examples include Autonomy, Ascential, divine, and Verity.

Gartner expects the second round of consolidation to begin before the conclusion of this year. Driving this consolidation will be the bundling of licenses and technology by the large, independent software vendors. This business strategy will be coupled with a technology approach – the "portal-enabled application server," the merger of the application server with basic portal services. As a result of these two trends, functionality requirements for future portal products will increase. The victims of this second round of consolidation will be more pure-plays and middle-tier software vendors.

Gartner's Four-Generation portal functionality model identifies a robust application framework as one of the key attributes of Generation-Two portals and their descendants. Either bundling an application server or running on top of a variety of application server platforms constitutes a key part of this framework. As the large, Java-centric platform vendors (e.g., IBM, Sybase, Oracle, BEA Systems, and Sun Microsystems) add basic levels of portal services to their Java 2 Enterprise Edition (J2EE) application servers, other portal vendors will be forced to enhance the functionality of their own products to remain competitive. The required technical expertise and resources will serve as barriers to many existing portal product vendors.

Although many application servers currently contain personalization features, they don't provide portal services. However, almost all application server vendors have a portal product. These vendors will "move the bar" between their application servers and their portal products, thus causing the portal product industry as a whole to change. By 2004, the majority of platform providers will include basic portal functionality in their application servers.

Over time, the platform vendors will separate their future portal products from their application servers, offering both the portal application server, including basic portal services, and a prime portal product delivering Generation-Three and Generation-Four functionality. The portal application server will provide enough functionality for a basic portal, while the prime portal product will serve up the "bells and whistles." The pure-plays and middle-tier vendors will be faced with a dilemma. They will be forced to enhance their portal products to go well beyond the basic portal services that will become bundled into portal-enabled application servers. This effort will drive many vendors out of the portal product market, or will force them to focus on price competition. These vendors will also have to start leveraging the built-in portal services provided in the portal-enabled application servers. Additionally, while the large, Java-centric platform vendors will leverage J2EE standards in a way that will result in consistent definitions of portal services, they will each create extensions. This will require the pure-plays and middle-tier vendors to write to each platform separately. Also, some pure-play vendors may take the opportunity to court the large platform providers.

Gartner believes that if an enterprise has already invested in the application server of a large platform provider, the enterprise should closely examine the future portal and application server strategies of that vendor. Enterprises seeking to deploy the portal product of a pure-play or middle-tier vendor should carefully examine the their strategies for evolving their products, and strategies for leveraging the features of the portal-enabled application servers of the large, Java-centric platform vendors. This advice is particularly important given the current

WebSphere Developer's Journal 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

WebSphere Developer's Journal 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

state of the portal product market and the large number of enterprises that will be moving from running departmental portals to deploying portals across the enterprise.

The portal product market is approximately four years old, but it wasn't until 2000 that large, enterprise-wide deployments began in earnest. Most enterprises that invested in portal products are still running departmental portals, or are in the process of deploying them across the enterprise. The three most common issues identified by enterprises deploying large-scale portals are scalability, the total cost of ownership (TCO) of portlets, and the consequences of the lack of portal governance.

Portal products that evolved from thin, Web-presentation architectures may be hidden time bombs. A thin Web-presentation layer will not grow into a mission-critical environment that scales to large numbers without some major re-architecting. Vendors who took the time to redesign their architectures along the way have a solid chance of deploying a mission-critical infrastructure. The products of vendors that incrementally changed their architectures, rather than redesigned them, will likely run into issues of reliability and scalability as portal use grows. Many users who invested in this latter set of vendors could see their portals "blow out" in 2002 and will be forced to abandon the original portal product and invest in a new, more robust one.

There are two major mechanisms for integration in the portal product market: prebuilt integration components (a.k.a. "portlets") and abstraction approaches. Portal products with a long list of portlets are very desirable because they provide a rich portal experience almost overnight. However, portal products that rely exclusively on the portlet model will be more costly to support over time because portlets are inherently a less-sophisticated architecture. In the words of one experienced portal developer, "portlets are brittle," reflecting the fact that as back-end systems and repositories change, the front-end portlets that access them must change at the same time, and breakage frequently occurs when changes aren't synchronized. In 2002, many users who invested in portal products that are heavy in portlets and light in abstraction will realize that the overall TCO of these portals is much higher due to portlets' brittle nature.

A lack of portal governance in large enterprises is common. Without appropriate governance, the possibility of multiple horizontal portals in a single enterprise is high. Because of the lack of interoperability standards, portal products from different vendors don't work together. Therefore, stovepipe portals are the result — which will become an increasing problem for large enterprises with autonomous business units/geographic units. While standards-creation efforts like JSR 168 are attempting to address these issues, Gartner doesn't expect a full range of portal interoperability standards to exist until at least 2004. This lack of interoperability standards will lead organizations to implement an "uberportal" as the only solution. The need for uberportals will become very clear in 2002, as will the difficulty in building them.

An uberportal is a lightweight, thin portal framework that overarches other portals. The uberportal is a horizontal portal, while the underlying portals may be horizontal or vertical. The key integration points between the uberportal and the underlying portals include directory, security, personalization profiles, metadata, and the portlets. The uberportal becomes the initial entry point, and user interface personalization will likely be handled there as well. An uberportal can't be bought "out of the box." It must be built on top of the framework of a horizontal portal product. The effort to tie an uberportal to its underlying portals is significant, so the need must be great to justify the effort.

Currently, the best positioned set of technologies to deliver on the hope for portal interoperability standards is Web services. Over the next 12 months, Web services will become the de facto standard for invocation of portlets. Vendors will have to wrap their portlets in Web services–compliant wrappers. This will be the first step for interoperability standards in the portal product market. Another aspect of Web services that will affect the portal product market is the user interface. Portals will likely be the genesis of emerging standards in this area. In fact, one of the pure-play vendors, Epicentric, has been pushing the envelope of Web services user interface standards with WSIA, which recently has been endorsed by IBM.

The portal product market has undergone some significant changes in the last year, and will continue to undergo further changes. In spite of the volatility of the market, Gartner feels that a strategic investment in a portal product is possible. Risk-averse companies should continue to invest tactically until late 2002. Enterprises that can accept and mitigate risk should feel safe in making strategic investments. However, extreme due diligence must be done.

"Over the next twelve months, Web services will become the de facto standard for invocation of portlets"

PRECISE

WWW.PRECISE.COM/WSDJ

MONGOOSE
TECHNOLOGY

WWW.PORTALSTUDIO.COM