WebSphere Portal: Unified user access to content, applications and services

by R. Will
S. Ramaswamy
T. Schaeck

Portals provide end users with unified access to content, applications, and collaboration services in a highly personalized manner. WebSphere Portal provides a middleware framework and tools for building and managing portals using component applications called portlets. In this paper we provide an overview of WebSphere Portal and describe its role in modern information technology systems. We describe its high-level architecture and main features, we detail some of its advanced features, and we outline directions for its future development.

A *Web portal*, or simply a *portal*, is a Web site that offers a broad array of resources and services typically targeted towards specific categories of user populations. Portals are rapidly gaining popularity and widespread adoption because they provide end users with unified access to applications, content, and collaboration services. Portals help corporate information technology (IT) staff by allowing them to integrate independently developed applications in a very cost-effective way. Portals also help site owners by allowing them to provide a consistent, branded experience to their user population while retaining control over individual user experience.

Portals have been around since the rapid adoption of the Internet started in the early 1990s. Early portals were built using homegrown frameworks and technologies and were targeted at users that required a single entry point to the Web. They allowed users to explicitly search for information or to browse a catalog. Examples of such portals that are still active are Yahoo!**, MSN**, AltaVista**, and Google.

In recent years, many enterprises have recognized the value of portals for streamlining and increasing the efficiency of their interactions with their constituencies. Consequently, a number of vendors started to offer products for building portals for the enterprise. The vendors offering such products include IBM, SAP, Microsoft, BEA, and Oracle.

WebSphere Portal[1] represents one of the most comprehensive portal frameworks in the industry. It provides customers with the wide range of tools and runtime capabilities required to implement advanced portal solutions. Many of the concepts first conceived and implemented in WebSphere Portal have been adopted as industry standards.

The remainder of this paper is organized as follows. We first explain the basic concepts and capabilities of WebSphere Portal, describe its operation, and present a brief overview of its internal structure. Next, we detail some of its advanced features that are included in the latest release. Finally, we discuss some of the newer concepts being explored for possible inclusion in future releases.

Overview

The WebSphere* software platform is concerned with three broad areas of functionality: (1) It pro-

0018-8670/04/$5.00 © 2004 IBM

Figure 1 Sample page produced by WebSphere Portal

vides access to applications, content, and collaboration services from a variety of devices, (2) it integrates and automates business processes, and (3) it builds, connects, and manages applications. WebSphere Portal is primarily focused on the first of these areas. It is the user-facing component of the platform, and its goal is to extend the "reach" of the platform by making WebSphere-hosted applications, services, and processes broadly accessible to any user on any device. In addition, WebSphere Portal can leverage IBM's Lotus*, Tivoli*, and data management products in order to provide the breadth and depth of services necessary for creating full-featured portal solutions. For example, Lotus collaboration services can be used to provide portal users with the ability to collaborate in various ways, Tivoli security services can be used to secure and manage the portal, and data management products can be used to provide the various types of storage facilities required by the portal.

WebSphere Portal operates by aggregating and integrating presentation from individual units of user experience (portlets) to create a unified experience for end users. The framework's value lies in making the process of creating and managing the unified experience simple and yet extremely flexible. WebSphere Portal's self-service features allow end-users to personalize and organize their own view of applications, to manage their own profiles, to share documents, and to collaborate with their colleagues. These aspects of WebSphere Portal contribute significantly to WebSphere as an on demand computing platform. Figure 1 illustrates a sample page produced by WebSphere Portal. This page contains six portlets laid out in three columns. The three port-

lets in the left column display a welcome message, a set of bookmarks, and a product synopsis (from top to bottom). The portlet in the center column displays a summary of the day's news. The two portlets in the right column display a list of stock quotes and weather information for a list of cities (from top to bottom). For detailed information on WebSphere Portal, see Reference 1.

Portal model and portlets. WebSphere Portal operation is based on a model for user experience, the *portal model*. The portal framework interprets the portal model in order to create the portal experience for end users. The portal model covers aspects such as the content hierarchy and associated navigation, security settings, user roles, customization settings, and device capabilities. A *portlet* is an application that underlies a window within a portal Web page. The portal model includes a scheme for the aggregation and integration of a set of portlets into a unified portal experience. For a given set of portlets, the portal model provides flexibility in terms of how they can be used; by changing the portal model within which they are used, different portal experiences can be obtained. For example, the sample page in Figure 1 is produced by using a page definition that specifies a three column layout, the portlets to be included in each column, and the order in which they are to be displayed.

Each portlet is an independently developed Web application using an extended J2EE** (Java2 Platform, Enterprise Edition) Web application architecture.[2] The extensions to this architecture ensure that multiple portlets can coexist on a single Web page, that portlets can be easily customized based on their role within the portal model, that portlets can work together to help users accomplish a given task, and that portlets can be properly rendered for a variety of user devices. For example, a portlet may create one or more *portlet instances*; a weather portlet might have an instance configured to show the local weather, while another instance of the portlet on another Web page of the portal might show the weather in the 10 largest cities of the world.

The portal provides a set of administrative functions that allow authorized users to collaboratively define the portal model, including the content hierarchy (pages, peer pages, nested pages, etc.), the content layout for each page (the number of columns or arbitrarily nested rows and columns), and the binding of portlets to specific sections of the page. As each page is defined, and a portlet is incorporated into the page, the administrator may identify the users who can view and modify the page and may specify the settings for the portlet. For example, the administrator may specify that only users in the Managers group can see the Personnel page or that only users in the Human Resources group can see the Hiring Projections portlet within the Personnel page. An important administrative capability is delegation. When defining pages and layouts, the administrator may lock down some sections of the portal model (prohibiting others from altering its content) and may delegate management rights for remaining sections of the portal model (such as pages or parts of pages) to other administrators or even end users. In this way, a company can centralize control of the content of some pages or parts of a page, while delegating control of the content of other pages to regional administrators within the company. The central administrator, or any of the regional administrators, can allow end users to customize pages or parts of pages according to their own preferences. Finally, an administrator or authorized user can specify the pages or parts of pages that should appear on any given device. In this way, information that is too "dense" to display well on some pervasive devices is omitted when the user accesses the portal from one of these devices.

After the portal model is defined, further customization is supported at the portlet level through *portlet modes*. WebSphere Portal defines four viewing modes. The View mode, which is the default mode, represents the typical view seen by users. The Edit mode is the view used to set "instance data" for a portlet instance. In our weather portlet, for example, the instance data would be the list of cities for which the portlet instance is to display the weather. The Configure mode is the view used by administrators to set values such as data sources for a portlet. Finally, the Help mode is the view for users to receive help information.

Themes and skins. The J2EE JavaServer Pages** (JSP**)[3] technology helps separate the presentation and the control functions of an application. WebSphere Portal takes this a step further with *themes* and *skins*, which isolate branding and look-and-feel aspects of the user experience by using a well-defined set of artifacts that include JSPs and CSS (Cascading Style Sheet)[4] style classes.

The theme is responsible for rendering all currently visible parts of the portal model. It determines how navigation links are to be rendered (for example, us-

ing nested tabs or using a tree that expands and collapses), where the actual content of the page is to be rendered, where logos and other commonly used links and services (for example, search or bookmarks) are to be surfaced, and what fonts and colors should be used. Skins are responsible for the decoration around individual portlets. Skins can surface controls for switching portlet modes or window states and display portlet titles.

Themes and skins are referenced within the portal model and used appropriately to produce the user experience. The branding and overall look and feel of the portal can be changed easily by simply replacing the themes and skins referenced in the portal model. Portlet programmers are expected to make use of theme-defined CSS style classes within their implementations. This allows independently developed portlets that are visible on the same page to look as if they were developed by a single team.

Content and documents. Aside from providing access to applications, access to content remains an important function of portals. WebSphere Portal provides a Web content management capability that allows users to easily create structured content for their portal. Structured content is separate from presentation (a database table row or XML [eXtensible Markup Language] documents are good models of structured content).

WebSphere Portal's content publishing function allows portal users to contribute structured content by filling in forms and submitting the content. This content is then stored in a relational database or DB2* Content Manager[5] and is displayed in one or more portlets. WebSphere Portal provides a great deal of flexibility as to the content that shows up in a portlet. Typically each portlet implementation can use a set of business rules to determine what content to show within a given instance. For example, one portlet might show news that is relevant to managers; at the same time, that same portlet on another page might show news that is of general interest. WebSphere Portal provides advanced business rules that allow fine-grained content targeting. For example, a News portlet might show news targeted to the user's locality or job responsibility and experience level.

In addition to typical Web content, WebSphere Portal also provides a portal document management capability. Portal document management provides the foundation for portal users to collaborate on the production and use of documents of various types. Simple document editors for popular document types (such as rich text, spreadsheets, and presentations) are also included with WebSphere Portal. These editors provide convenient access to basic functions and are not meant to replace the more sophisticated stand-alone products.

Another important content-related capability in any portal is the ability to search for relevant information. WebSphere Portal provides advanced search capabilities, including automatic categorization and summarization of portal content and documents. It also allows for the integration of other search technologies in order to enable federated searches.

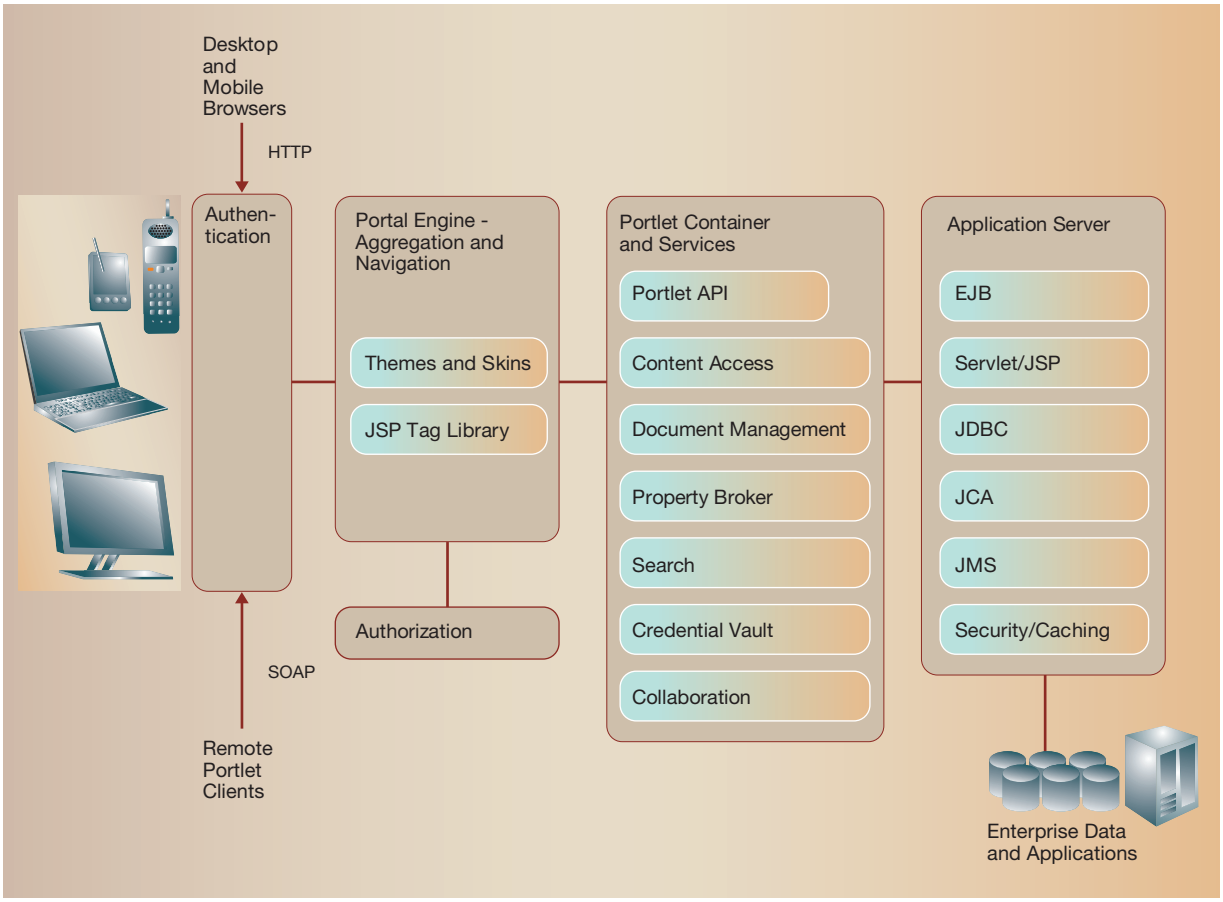
High-level architecture

Figure 2 illustrates the high-level architecture of WebSphere Portal. The portal has two primary access protocols: HTTP (HyperText Transfer Protocol) for client browsers and SOAP (Simple Object Access Protocol) for remote portlet clients via the Web Services for Remote Portlets (WSRP) standard.[6,7] For secure access to the portal, the first step is to verify the user's identity. WebSphere Portal uses WebSphere Application Server's authentication services to verify the user's identity. The *single sign-on support* allows the user, after logging on, access to all applications within the realm of the portal. For applications outside the single sign-on support, WebSphere Portal includes a credential vault (e.g., Tivoli credential vault) that can be used to store user sign-on information.

The authorization component determines the permissible actions within the portal for an authenticated portal user. It is the single control point within the portal that controls access to all parts of the portal model, such as pages or portlets. The authorization component is used by various other portal components—they allow actions on specific resources only if these actions are allowed by the authorization component. For example, the administrative components use it to determine the pages that users may customize and the portlets that they may use. Also, the aggregation module uses it to check whether the requested content may actually be displayed for a given user.

The authorization component in WebSphere Portal supports access control configuration of resource hierarchies through the concept of *permission inheritance*. This concept reduces the administration ef-

Figure 2 High-level architecture of WebSphere Portal



fort when controlling access to a large number of hierarchical portal resources. Inherited permissions are automatically assembled into roles that can be assigned to individual users and user groups, granting them access to entire sets of logically related portal resources. The user-to-role assignments can be managed within the portal or within external authorization systems (e.g., Tivoli Access Manager).

The portal engine implements aggregation of portlets based on the underlying portal model and information such as security settings, user roles, customization settings, and device capabilities. Within the rendered page, the portal automatically generates the appropriate set of navigation elements based on the portal model. Thus, there is no need for the site administrator to maintain navigation elements as the site changes and no chance for broken links.

The portal engine invokes portlets as and when required during the aggregation and uses caching to reduce the number of requests made to portlets. WebSphere Portal employs open standards such as the Java Portlet API[8] (application programming interface). It also supports the use of the remote portlet via the WSRP[6,7] standard. The portal allows for manipulation of requests and responses by supporting *filters* at two levels—*servlet filters* can be used to modify requests and responses at the portal engine level and *portlet filters* can be used to modify requests and responses of individual portlets. Portlets running in the WebSphere Portal environment can utilize a whole host of prebuilt portlet services as well as custom-built portlet services (described later). For example, the prebuilt *property broker* service (again, described later) can be used for creating brokered portlet integration.

The *portlet container* provides the runtime environment for portlets. Beyond the capabilities that a servlet container provides, such as life-cycle management and request processing, the portlet container provides facilities for event handling, interportlet messaging, access to portlet instance and configuration data, among others. The portlet container is not a stand-alone entity like the servlet container. As portlets are an extension of the servlet programming model, the portlet container uses servlet container functions and only adds portlet-specific functionality.

The Portlet API introduced by WebSphere Portal served as a model for the recently standardized Java Portlet API. It defines the interfaces and contracts between the portlet container and portlets. IBM and Sun Microsystems, Inc. were co-leads in the standardization of the Java Portlet API. Although many advanced aspects of the WebSphere Portal Portlet API are not yet part of the Java standard, it is expected that these will evolve into standards over time. In the meantime, WebSphere Portal will support both proprietary and standard portlet APIs and will allow interoperability between portlets developed using either API.

Portlet services are a means of introducing common functions that can be shared across portlets. WebSphere Portal provides a set of prebuilt services such as the *content access service*, the *property broker service*, and the *document management service*. Additional custom portlet services can be defined and implemented as necessary. In order to be properly instantiated, these custom portlet services must be registered with the portal. Portlet services are initialized only upon demand. Each portlet service implements a custom interface that portlets can use to access its function. Note that portlets call into portlet services during the request processing cycle and pass in appropriate context for the portlet service to provide its function. Portlet services do not initiate calls into portlets.

Advanced features

WebSphere Portal provides many advanced capabilities that are unique in the marketplace. For example, independently developed portlets can share information via the property broker in WebSphere Portal (described in more detail later). This allows for low-cost integration in the front end, at the presentation tier. WebSphere Portal allows a portlet developer to specify the values that a particular portlet can produce and the values that a particular portlet can consume. For example, an application can return shipping information for a given order, whereas a second application can return order information for a given customer. Using tags, APIs, and administrative functions, these applications can be wired together so that the orders for a given customer can be located, and the shipping information for these orders can be automatically displayed.

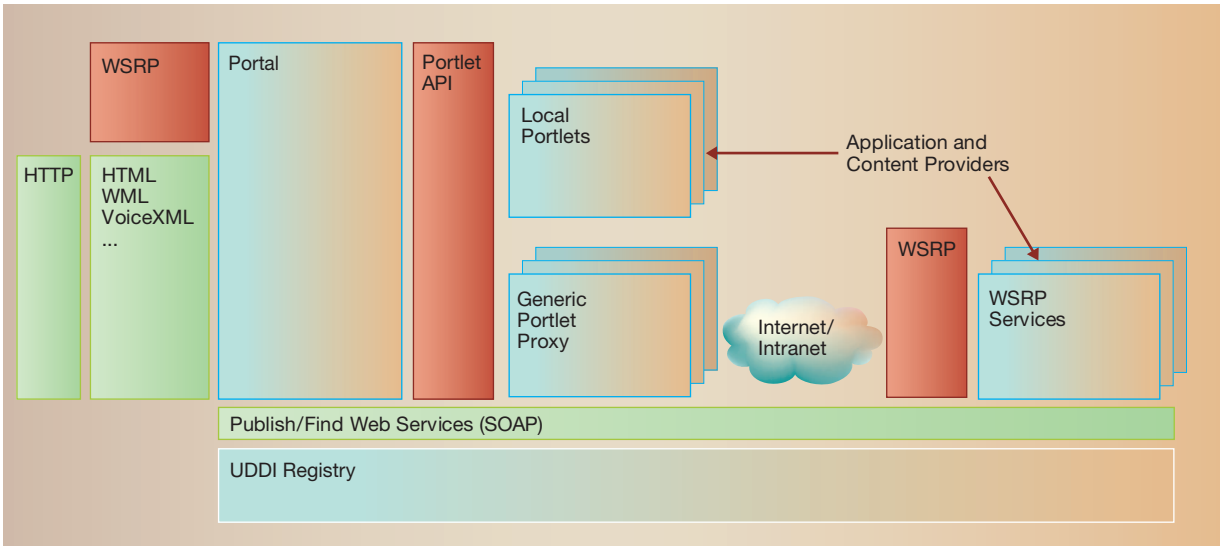
Web services and WSRP. Although portlets can use Web services to implement their function, this requires a significant programming effort. The WSRP standard simplifies integration of remote applications and content into a portal via Web services without programming.[6,7] WSRP standardizes Web services at the presentation layer on top of the existing Web services stack, builds on the existing Web services standards, and will benefit from additional Web services standards efforts (such as the security work now underway) as they become available.

At its core, the WSRP standard defines pluggable, user-facing, interactive Web services that implement Web Services Description Language (WSDL)[9] interfaces and a fixed protocol for processing user interactions and providing presentation fragments suitable for mediation and aggregation by portals. WSRP defines the conventions for publishing, finding, and binding such services. WSRP permits any compliant Web service to be plugged into any compliant portal without requiring any service-specific adapters—a single, generic adapter on the portal end is sufficient.

WebSphere Portal supports the WSRP standard both from the consumer perspective and the producer perspective. In Figure 3, the red box on the right side of the Internet cloud labeled WSRP depicts the consumer perspective. Given that all WSRP services are identical, a generic WSRP proxy portlet is used to consume all WSRP services. This generic WSRP proxy portlet is written to the Java Portlet API (depicted by the red box labeled Portlet API) and runs in the portlet container along with other portlets. Portal administrators can publish any portlets (written to the Java Portlet API) as WSRP services for use by other portals, as depicted by the red box labeled WSRP on the left side of the figure.

Brokered integration. As mentioned earlier, the portlet container in WebSphere Portal supports interportlet messaging. This can be used to achieve "front-end" integration—user interactions with one portlet on the page are reflected in others on the

Figure 3 Java Portlet API (JSR 168) and WSRP employed in WebSphere Portal



page. For example, a user looks up a sales order in one portlet. Using interportlet messaging, the customer ID field from the sales order is sent to another portlet on the page; the second portlet uses it to display customer information relevant to the sales order. We point out, however, that using messaging directly to achieve front-end integration requires portlets to be developed using a collaborative development process. Furthermore, future changes in one portlet may cause unexpected behavior in the other portlet, requiring a collaborative development process for maintenance as well.

To overcome these drawbacks, we introduce the concept of a brokered front-end integration mechanism called the *property broker*. From a historic perspective, this development can be compared with back-end integration. The first methods of back-end integration involved applications exchanging messages, which meant that the integration logic resided in the applications themselves and was therefore cumbersome to track and maintain. Current methods of back-end integration, however, involve a brokered architecture where the broker contains all the integration logic; that is, the applications interact only with the broker and not with each other.

The property broker relies on portlets being producers and consumers of typed properties. Portlets can either register directly as consumers and producers of properties or indirectly via specific actions that consume and produce properties. The property broker facilitates interactions between portlets either by allowing the property produced by a portlet to be consumed by another portlet or by allowing a property produced by a portlet to trigger an action on another portlet. The property broker uses the types associated with properties to determine compatibility between properties belonging to different portlets.

The property broker does not by itself orchestrate the interactions between portlets. Instead, it allows portal users to manually direct interactions by presenting them with a means to trigger any of the valid interactions (Click-To-Action). It also allows portal administrators to specify the automatic triggering of interactions with page definitions (cooperative portlets).

WebSphere Portal Application Integrator. The WebSphere Portal Application Integrator (WPAI)[10] provides rapid integration of business data and associated functions from popular back-end applications into WebSphere Portal. We use the term back-end applications to denote any of the following:

- Enterprise application suites from Siebel, SAP, PeopleSoft, and so forth.

- Relational databases such as IBM DB2, Oracle Database 10g, and Microsoft SQL Server.
- Business intelligence applications, such as IBM DB2 OLAP (On-line Analytical Processor) Server, and SAP Business Warehouse.
- Other applications such as e-mail and Lotus Domino*, and enterprise solutions such as those provided by Ariba, Inc. and Documentum, Inc.

WPAI provides support for rapidly creating application portlets for back-end applications by using a template-driven approach. There are two key concepts underlying this approach. The first concept is the classification of models representing data from back-end applications into model classes whose members can be accessed via a common interface. Model class adapters are created for various back-end applications that utilize a back-end-specific interface to implement the model-class-specific interface. Examples of model classes are business objects that correspond to business entities such as sales orders, customer records, employee records, and performance ratings. Other model classes are e-mail, OLAP cubes, workflow processes, Domino forms and views, structured content, and document repositories. The second concept is the development of user interface templates that implement model-class-specific patterns of user interactions by using controllers and views developed to the associated model-class interface. An example of a user interface template is one that allows users to search and browse through instances of a business object type by following these steps:

- Use a search input view to enter some search criteria appropriate to the business object type, and press a button to perform the search.
- View the set of matching business object instances with only a selected subset of information for each instance being presented, and select a particular instance of interest.
- View complete details of the selected business object instance.

Application portlets for a model class are now created by combining a desired user interface template for the model class with a desired model obtained from the appropriate model class adapter. For example, an application portlet for the Business Objects model class may be created by combining a search-and-browse user interface template for the Business Objects model class with the SAP sales-order business object obtained by using a SAP business-object adapter. This process is illustrated in Figure 4.

WPAI is shipped with WebSphere Portal and currently includes support for the business-object model class (from SAP, Siebel, PeopleSoft, and various databases) and the Lotus Domino* forms-and-views model class through appropriate adapters and user interface templates. It also includes portlet-builder tools that facilitate the assembly of user interfaces and models from adapters into the application portlet. Support for other model classes will be made available in future versions of WebSphere Portal.

Future work

We discuss here three ideas that might be implemented in future version of WebSphere Portal: dynamic assembly, virtual portal, and process portal.

Dynamic assembly. WebSphere Portal is based on a flexible portal model that is currently set up statically (for example, by using the administrative functions). In order to support scenarios in which the data to be presented to the user is determined dynamically (e.g., based on factors external to the system, on personalization rules, on changing marketing campaigns, or on user actions), WebSphere Portal will support the concept of *dynamic assembly*.

Using this concept, administrators can attach dynamic-assembly transformations to any element in the portal model. When a dynamic element with an attached transformation is encountered during the rendering of the portal model, the transformation gets triggered and can dynamically determine the section of the portal model that lies beneath the dynamic element. Transformations are provided with a user-scoped section of the portal model that lies beneath the dynamic element and may filter the model or add nodes or entirely new sections. This is illustrated in Figure 5. The large triangle represents the entire portal model (depicted using a tree structure), and the dark node in the tree represents a node with which a dynamic-assembly transformation is associated. When the node is encountered by the portal framework during rendering, it invokes the associated transformation, providing as input to the transformation the subtree at which the dark node is rooted (depicted by the small triangle enclosed within the large triangle). The transformation process itself can choose to modify the subtree in whatever fashion is deemed to be appropriate, and the modified subtree is now used in place of the original subtree. This is depicted in the figure by the three triangles above the large triangle. For example, dynamic assembly could be used to determine the set of pages that ap-

Figure 4 Templates and adapters used for rapid creation of portlets

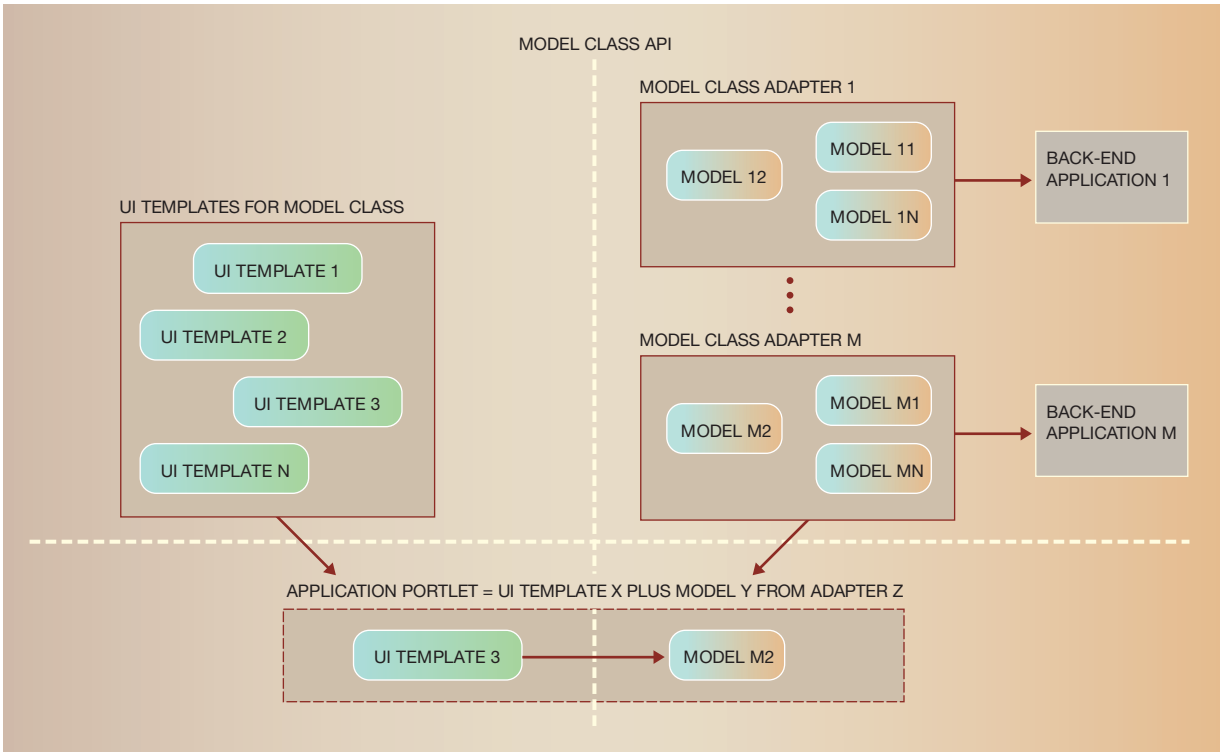
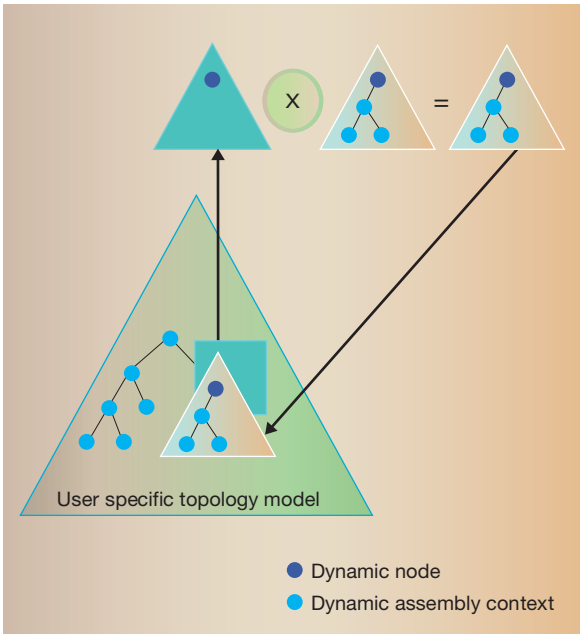


Figure 5 Dynamic assembly transformations



pear under a given page based on personalization rules. This would allow the set of navigation options and content to be tied to user attributes such as sex or marital status.

Virtual portals. While companies have initially deployed individual portals, the trend is towards consolidation of portal installations for better utilization of the hardware running portals and for increased operational efficiency. WebSphere Portal intends to support *virtual portals* as first-class entities in the portal model, which means hosting multiple logical portals within a single physical portal installation. Virtual portals enable on demand computing by facilitating the addition and removal of portals. Furthermore, adding portals merely results in changing the portal model; it does not increase the footprint of installed code and does not require additional runtime resources.

Each virtual portal will have its own URLs (uniform resource locators), its own subset of users registered in the overall portal user registry, and its own content hierarchy and content. Portlets can be shared

between virtual portals. The delegated administration model described earlier will be extended to support virtual portals.

Process portal. The *process portal* capability is intended to enhance the productivity of WebSphere Portal users by providing support for carrying out tasks assigned to them. Users will be alerted when there are tasks waiting for their action via the most appropriate means to ensure prompt attention. For example, users may see a flashing alert when accessing the portal or receiving a text page. Users have access to a consolidated list of tasks that are appropriately prioritized. On selecting a task, the portal automatically navigates to the designated page for performing that task, and users are able to launch and deal with multiple tasks simultaneously.

With the process portal feature, the portal can be the single point of interaction for users so they can seamlessly initiate, monitor, and work with the business processes for a given enterprise. When performing nontrivial tasks, a user will often require additional input to make decisions. This includes input not only from other systems but also from other human beings. Because the portal is designed to be the single point of access to either of these resources, it presents an ideal environment for users to perform process-related tasks.

Summary

WebSphere Portal is a powerful framework for providing users with a seamless and customized experience when accessing applications and content from a wide and disparate set of sources. It supports industry standards, and through an advanced set of features, it reduces the complexity of integrating applications and content. The WebSphere Portal team continues to innovate in many areas, including brokered front-end integration, process integration, virtualization, and dynamic assembly.

Cited references

1. *WebSphere Portal for Multiplatforms*, IBM Corporation, http://www.ibm.com/software/genservers/portal/.
2. *Java 2 Platform, Enterprise Edition*, Sun Microsystems, Inc., http://java.sun.com/j2ee).
3. *JavaServer Pages Technology*, Sun Microsystems, Inc., http://java.sun.com/products/jsp.
4. *Cascading Style Sheets*, World Wide Web Consortium (W3C), http://www.w3.org/Style/CSS).
5. *IBM DB2 Content Manager*, IBM Corporation, http://www.ibm.com/software/data/cm.
6. T. Schaeck and R. Thompson, *Enabling Interactive, Presentation-Oriented Content Services Through the WSRP Standard*, Organization for the Advancement of Structured Information Standards (OASIS) (2003), http://www.oasis-open.org/committees/download.php/3657/WSRP_paper.html.
7. *Web Services for Remote Portlets Specification*, Organization for the Advancement of Structured Information Standards (OASIS) (2003), http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf.
8. *Java Portlet API Specification (JSR 168)*, Java Community Process (JCP), http://www.jcp.org/en/jsr/detail?id=168.
9. *Web Services Description Language (WSDL) 1.1*, World Wide Web Consortium (W3C), http://www.w3.org/TR/wsdl.
10. *IBM Application Portlet Builder (WPAI)*, IBM Corporation, http://www.ibm.com/developerworks/websphere/zones/portal/catalog/doc/appportletbuilder/ApplicationPortletBuilder_v410_wsdd.html.

Accepted for publication February 2, 2004.

Robert Will *IBM Corporation, 4205 S. Miami Blvd, Durham, NC 27703 (willrc@us.ibm.com).* Rob Will, a Distinguished Engineer in IBM's Software Group (Lotus Division), is the chief architect of WebSphere Portal. He received a B.S. degree and an M.S. degree in mathematics from Auburn University in 1979 and 1981, respectively. Rob has been a member of the WebSphere product development team since the beginning and has worked also on both WebSphere Application Server and WebSphere Studio. Most recently Rob has been working on WebSphere Portal, Web Content Management, and Document Management. Before WebSphere, Rob worked in software strategy, DCE development, System/390® Client/Server development, and VM/ESA® development.

Shankar Ramaswamy *IBM Corporation, 4205 S. Miami Blvd, Durham, NC 27703 (shankar4@us.ibm.com).* Dr. Ramaswamy is a Senior Technical Staff Member in IBM's Software Group (Lotus Division) working on WebSphere Portal architecture and development tools. He received a B.Sc. degree in electronics from the University of Delhi in 1987, an M.E. degree in electrical engineering from the Indian Institute of Science in 1991, and a Ph.D. in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1996. Since graduation, Dr. Ramaswamy has worked on several middleware products and technologies, including DCE/Encina® and WebSphere Application Server. He is the author of several research papers and patents, and he has won several academic and professional awards during his career.

Thomas Schaeck *IBM Research and Development Lab Boeblingen, Boeblingen, 71032 Germany (schaeck@de.ibm.com).* Thomas Schaeck, a Senior Technical Staff Member in Lotus Portal Development, is the architect for the WebSphere Portal Foundation. In parallel to working on the product architecture for WebSphere Portal, he initiated the Java Portlet API and WSRP standards and the Apache Pluto and WSRP4J open-source projects as reference implementations of these standards. He was the first chair of the OASIS WSRP technical committee and now leads portal standards activities in IBM.