Portal Server Technology

Christian Wege • *DaimlerChrysler and University of Tübingen* • *wege@acm.org*

To provide a consistent experience for customers, most Web site developers must integrate existing content with new content, server-side applications, and Web-based services. Increasingly, even basic sites are becoming more like traditional portals — a single, integrated point of access to information, applications, and people. Portals integrate diverse interaction channels at a central point, providing a comprehensive context and an aggregated view across all information.

Portals are largely based on existing Web application technology, such as Web servers and Java 2 Platform Enterprise Edition (J2EE). Here, I offer an overview of portal types and services, followed by a more detailed examination of portal-specific components and architectures.

Overview: Portal Usage and Services

Portals vary according to the users they serve and the services they offer.

- *Public portals*, such as Yahoo, are generally available and bring together information from various sources, applications, and people, offering personalized Web sites for arbitrary users (see Figure 1, next page).
- *Enterprise portals* (or "corporate desktops") give employees access to organization-specific information and applications.
- *Marketplace portals*, such as eBay and ChemWeb, are trading hubs that connect sellers and buyers.
- *Specialized portals*, such as the SAP portal, offer an access path to specific applications.

The "Portal Resources" sidebar on page 77 offers URLs for different portal types.

Despite the different usage scenarios, all portal types share a few common features. Portal server technology, still waiting for portal-specific standards (see the sidebar, "Portal Server Standardization Efforts on page 76), aims to provide portal implementers with a common set of services:

- *Customization* recognizes different users and offers them specific content configured to their needs. The service is based on gathering information about users and user communities and delivering the right content at the right time.
- *Content aggregation* prepares content from different sources for different users. It considers the user-specific context through the security service's authentication call, and the customization service's personalization call.
- *Content syndication* gathers content from different sources. Generally, the syndication service talks to every attached back-end system via the appropriate protocol. Professional content providers often make content available in standardized formats, such as rich site summary (RSS), news industry text format (NITF), and NewsML, an XML-based standard used to represent and manage news through its life cycle. Often, the quickest solution is to "clip" content from existing Web sites by copying HTML content into the portal. An employee portal, for example, might clip content from the corporate intranet.
- *Multidevice support* prepares content for different interaction channels, such as those for wired and wireless phones, pagers, and faxes, by considerng their characteristic capabilities. This typically requires a transcoding service to filter content by, for example, removing all images for a wireless phone and translating the HTML to wireless markup language (WML).
- *Single sign-on* services let the syndication service access back-end systems and retrieve user-specific information without requiring user authentication each time. The number of systems that require authentication and want to become accessible via a portal is growing

 1089-7801/02/$17.00 ©2002 IEEE

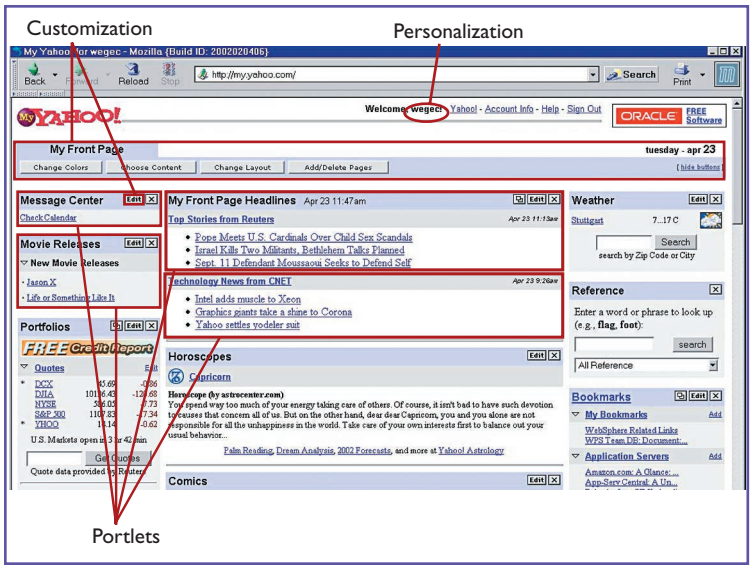

Figure 1. A page from the Yahoo public portal. The portal's pages feature user-specific information offered through customized portal components.

```
import org.apache.jetspeed.portal.portlets.AbstractPortlet;
import org.apache.jetspeed.util.MimeType;
import org.apache.turbine.util.RunData;
import org.apache.ecs.*;

public class HelloWorldPortlet extends AbstractPortlet
{
  public ConcreteElement getContent (RunData aRunData)
  {
    return (new StringElement ("Hello World!"));
  }

  public boolean supportsType (MimeType clientMimeType)
  {
    if (MimeType.HTML.equals(clientMimeType)) return true;
    if (MimeType.WML.equals(clientMimeType)) return true;
    return false;
  }
}
```

Figure 2. A simple portlet. A portlet implementation, such as this one for Apache's Jetspeed framework, will be instantiated once in the portlet container and then shared between multiple threads.

rapidly; applications for corporate human resource services is one example.

- *Portal administration* determines how users see the portal. This is more than look-and-feel; administrators must define user groups, interaction channels, and authorization information as well, depending on the portal's nature.
- *Portal user management* varies depending on the portal's audience. Users can typically subscribe themselves to public Web portals, for example, but not to enterprise portals. Also, depending on the portal type, the number of users can vary from several dozen to tens of thousands to millions. In some cases, administrators must categorize portal users into groups, so that the portal can present content specific to a user's role, interests, location, function, or position.

Portal-Specific Components and Architecture

To implement common services, many portal server implementations use similar architectural concepts, including portlets, portal server architectures, and portal-integration with remote portlets.

Portlets and Portlet Containers

Content providers make content available to users as *portlets* — content containers that are basically the users' view of their customized content (see Figure 1). Technically, a portlet is a piece of code that runs on the portal server and provides content to be embedded into portal pages.

I'll explain portlets and portlet containers in terms of J2EE concepts, since J2EE supports both portlet design and portlet interaction with run-time environments. Also, most portal server implementations are J2EE-based Web applications.

J2EE is one of the most widely known models for making components available on a server. In J2EE, server-side components live in specialized containers. The container is the server-side components run-time environment; it calls the component and provides component-specific services (such as user information and persistence service).

Java Server Pages (JSP) live in a JSP container, for example, and servlets live in a servlet container. Likewise, portlets live in portlet containers. The portlet API defines the interface between the portlet and the portlet container.

Figure 2 shows a simple portlet implementation. Like a servlet, a portlet is a singleton that is instantiated once in the portlet container, then shared between multiple threads. Important elements of the portlet API include:

- *request and response*, which provides and receives information specific to the portlet invocation;
- *session*, which stores information across portlet invocations;
- *config objects*, which contain configuration information about the user and portlet context; and
- *actions*, which enable interaction between mul-

tiple portlets, by implementing a publish–subscriber-like interaction model.

Portlets are available in several modes. Users can view present content, launch help for a particular view, or edit the view to customize it to their preferences, and administrators can configure the portal to customize services. The mode users select determines which portlet interface they'll see. Orthogonally, the view can be in one of several states, including `normal`, `maximized`, `minimized`, `closed`, and so on. Like servlet deployment descriptors, portlet descriptors contain deployment-related properties for each portlet.

In addition to the portlet container, a portal server must provide portlet services such as *syndication services*, which cache content from unreliable content providers, and *persistence services*, which let portlets store information to a persistence medium. However, the most important portlet service is the *user info service*, which gives portlets access to user-related information including preferences, customization information, and so on.

Figure 3 shows the typical building blocks of a portal server built as a servlet application. The portal engine receives the servlet request from the servlet container and transforms the request into a portlet request that it dispatches to the appropriate portlet. The portlet must retrieve the content using the portlet services provided by the portal server. The portal engine then aggregates the multiple portlet response and returns a servlet response to the user. To render the page appropriately, the aggregation must account for user preferences and device capabilities.

As the "Portal Server Standardization Efforts" sidebar (next page) describes, a standard portlet API is in the works.

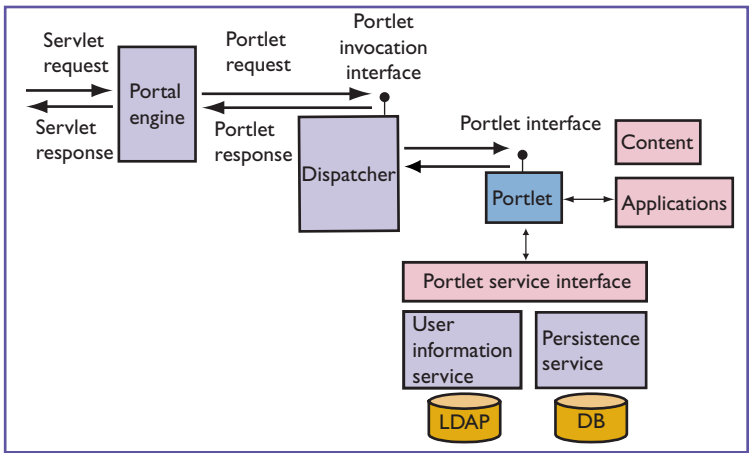

Figure 3. Building blocks of a portal server. This server is built as a servlet application, which often transforms a servlet request into many portlet requests and dispatches them to the appropriate portlets.

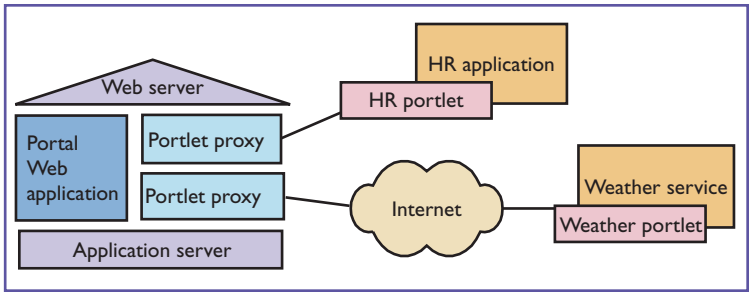

Figure 4. Portlets run on the local portal server. In this case, an employee portal provides users access to an Internet-based weather service and a human-resources application from an internal resource-planning system.

Remote Portlets

Portals draw information and content from many sources, including in-house systems and Internet-based content and application providers. To integrate a new source, portal providers must adapt the content to a format the portal understands, which can be a time-consuming and cumbersome process.

Currently, portal providers can reduce their integration effort in two ways. In the first approach, external service providers deliver content in a format that is directly consumable by a clipping portlet – typically HTML or WML. However, this approach limits the portal's ability to prepare content specific to the user's interaction channel. In the second approach, the portal provider installs a portlet in the external service provider's portal server. The portlet then consumes the content in its specific format to render it as part of the overall portal page. Putting third-party code on the portal server, however, can open both stability and security holes. For example, an employee portal might provide access to Internet-based weather information and a human-resources application in the corporate enterprise resource-planning system. To provide such access, administrators run the portlets locally on the portal server, as Figure 4 shows.

Remote portlets use intermediary proxies that let providers offer content or applications without requiring manual adaptation. Rather than plugging the portlet itself into the portal server, portal providers simply plug in a generic portlet proxy, which talks to the remote portlet. Remote portlets are hosted by another portal server or by a *portlet*

Portal Server Standardization Efforts

There are ongoing standardization efforts in two important areas of portal server technology: the portlet API and Web Services for Remote Portals

Portlet API

Experts within the Java Community Process are currently extending the J2EE standard within a Java Standardization Request (http://www.jcp.org/jsr/detail/168.jsp), with the goal of unifying several open-source projects and products from different portal vendors. In addition to unification, the group wants to ensure that the portlet API will be based on open standards.

The specification will comprise the portlets themselves, the portlet API, and the deployment descriptor format. Currently, however, there is no publicly available specification draft and no reference implementation.

Given that Apache is a member of this group, its Jetspeed project is likely to provide a reference implementation. However, the involved portal server vendors are expected to adopt standards more quickly than they'll adopt the open source implementation.

Web Services for Remote Portals

A technical committee from the Organization for the Advancement of Structured Information Standards is currently developing a standard for Web services for remote portals. WSRP will allow plug-and-play interoperation of visual, user-facing Web services with portals and other intermediary Web applications. It will also let providers implement remote portlets in any technology, whether it is J2EE, .NET, or any SOAP-accessible service.

The Oasis effort is supported by several content providers and most major portal server vendors, some of whom have products that already implement remote portlet capabilities. As with the portlet API effort, this standard will unify a fragmented area and ensure a specification based on open standards.

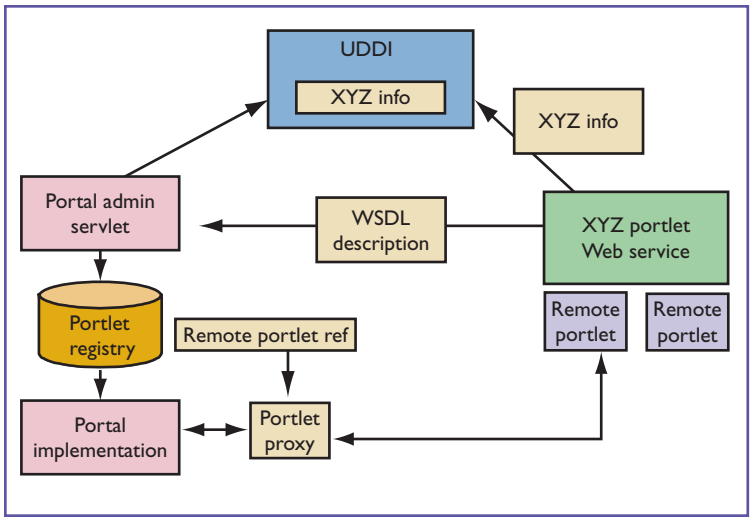

Figure 5. Locating remote portlets. The portal server finds remote portlets by searching the UDDI registry and calls the remote portlets using the portlet proxies.

runner, which is a stripped-down portal server that acts solely as a portlet execution environment. The portlet runner contains just enough functionality to enable the remote portlet to respond to the portlet proxy's calls. Thus, the portlet proxy does not need services such as aggregation and customization. The portlet runner can also be implemented in another technology (like .Net), as long as it can respond to the portlet proxy's remote portlet invocation protocol.

To enable interoperability between different portal servers and content providers, we need a standardized interaction model for the portlet proxy and the remote portlet. As the "Portlet Server Standardization Efforts" sidebar describes, the Organization for the Advancement of Structured Information Standards (Oasis) is currently working on a remote portlet Web services standard.

Web Services for Remote Portals

Web services for remote portals (WSRPs) are visual, user-facing Web service components that plug-and-play with portals or other intermediary Web applications that aggregate content from different sources. Content or application providers implement their service as a remote portal Web service and publish it in a globally accessible directory (such as the Universal Description, Discovery, and Integration registry — UDDI). The directory lets portal providers easily find a desired service. Directory entries, published in Web Services Description Language (WSDL) format, briefly describe the WSRP component and offer details about the services. The portlet proxy binds to the WSRP component through SOAP, and the remote portlet invocation (RPI) protocol ensures the proper interaction between both parties. Figure 5 shows an example of how a portal finds and integrates a remote portlet.

For more on WSRP, see www.ibm.com/developerworks/library/ws-wsrp and oasis-open.org/committees/wsrp.

Conclusion

Portal server technology is still a young and frag-

Portal Resources

Despite high expectations for portal server technology, it has yet to make much impact in the literature. Following is a list of existing publications, along with online resources mentioned in the article.

Online

- Apache Jetspeed Portal Framework • jakarta.apache.org/jetspeed/
- IBM WebSphere Portal Server • www.ibm.com/software/webservers/portal/
- BEA WebLogic Portal • www.bea.com/products/weblogic/portal
- Epicentric Foundation Server • www.epicentric.com
- SAP Portals • www.sapportals.com
- Plumtree Corporate Portal • www.plumtree.com
- ChemWeb portal • www.chemweb.com
- Yahoo portal • www.yahoo.com
- eBay portal • www.ebay.com

Books and articles

- Beehive and Casey Duncan, eds., *The Book of Zope*, Linux Journal Press, Seattle, 2001.
- Edd Dumbill, "XML at Jetspeed," XML.com, 2000; available at www.xml.com/pub/a/2000/05/15/jetspeed.
- Clive Finklestein and Peter Aitken, *Building Corporate Portals with XML*, Osborne McGraw-Hill, New York, 1999.
- Michele Galic et al., *Access Integration Pattern Using IBM WebSphere Portal Server*, IBM ITSO, 2001; available online at ibm.com/redbooks.
- Jeff Linwood, "Build Portals with Jetspeed," JavaWorld, 2001; available at www.javaworld.com/javaworld/jw-07-2001/jw-0727-jetspeed.html.
- Avi Saha, "Application Framework for e-Business: Portals," IBM developerWorks, 1999; available at www-106.ibm.com/developerworks/web/library/portals/index.html?dwzone=web.
- Thomas Schaeck and Stefan Hepper, "Portal Standards, The Server Side," 2002; available at www.theserverside.com/resources/article.jsp?l=Portlet_API.

mented field, but specific standards are in the works. There are, however, a few mature products on the market that define the current state-of-the-art.

Most portal server products today are Java/J2EE-based. Epicentric and Plumtree were among the first companies to offer portal servers as products. With the acquisition of TopTier, SAP is poised to join them because of its iViews integration layer for applications. IBM offers WebSphere Portal Server, which has its roots in Apache's Jetspeed open-source portal framework. Apache is involved in developing specifications for the J2EE standard's portlet API, which makes Jetspeed a strong candidate for becoming the upcoming standard's reference implementation. Another open-source portal framework is Zope, which is implemented in Python. Beyond portal functionality, Zope provides some content management and general Web application services. This list is by no means complete. The Java specification request (JSR) and Oasis specification provide a more comprehensive list of submitting and supporting companies (for more on this, see the "Portal Server Standardization Efforts" sidebar).

In the near future, we should see stabilization and increased interoperability between different vendor's portal servers as they adopt portal-specific standards. Including the portlet API as part of the J2EE specification will also boost the use of portal server technology significantly, given the J2EE specification's prominence. As with other elements of the J2EE specification, open-source implementations should soon be in wide use, challenging portal server vendors to maintain their competitive advantage. In the mid-term future, the rate of technology uptake will determine the viability of a portlet service market. Finally, in the long run, portal server technology has the potential to sneak into new areas. Imagine, for example, a printer that provides its administration user-interface as a remote portlet Web service inside a telephone-accessible systems management portal.

Christian Wege is a doctoral candidate at University of Tübingen, Germany, and an applications architect at DaimlerChrysler. He received a diploma in computer science from the University of Tübingen. His research interests include software architecture and interdependence between software evolution and process evolution in agile methodologies.