The Online Knowledge Center: Building a Component Based Portal

Ozgur Balsoy[1,2,3], Mehmet S. Aktas[2], Galip Aydin[2], Mehmet N. Aysan[2], Cevat Ikibas[2], Ali Kaplan[2], Jungkee Kim[2], Marlon Pierce[2,3], Ahmet Topcu[2], Beytullah Yildiz[2], and Geoffrey Fox[2]

[1]Computer Science Department, Florida State University
[2]Community Grid Labs, Indiana University

Postal Address: Community Grid Labs
501 N. Morton Street, Bloomington, IN 47402
Email Addresses: {obalsoy,maktas,gaydin,maysan,cikibas,alikapla,kimjungk,mpierce, atopcu,byildiz}@indiana.edu

[3]Corresponding Authors
Telephone Number for Corresponding Authors: (812) 856-0751, (812) 856-1212
Fax Number for Corresponding Authors: (812) 856-7972

Submitted to Information and Knowledge Engineering '02 as RRR

Abstract

This paper presents an overview of the Online Knowledge Center (OKC) web portal. The OKC is built around a portlet/container architecture: a central control portal is composed of several portlets that can deliver both local content and content from remote servers. The modular structure allows us to develop sophisticated portal components independently and plug them into the portal container using well defined XML interfaces. We describe problems we have discovered with this architecture and extensions and solutions that we are implementing. We also describe two advanced services that can be plugged into the overall framework: XML message-based newsgroups and hybrid structured/unstructured data searches.

1. Introduction

The Online Knowledge Center (OKC) is a component based portal system that we are designing to support distributed web content display, management, and authoring. The OKC will potentially be used to provide access to a wide range of information in various formats, including but not limited to presentation slides, software repositories, contact information, training announcements, and newsgroup postings.

Component-based portals represent an important area for investigation and development. Essentially, this allows us to treat all web accessible content as objects that can be wrapped inside standard XML interfaces. Through a standard web service framework such as WSRP[1], portal containers will be able to dynamically discover and bind to desired content portlets. We view this as the correct interface management system for user interfaces to web services: portals become an aggregate of distributed portlets, each of which in turn is a client to one or more web services. All components describe themselves and communicate with XML.

In summary, the OKC development and research focuses on the following major areas: portal components that manage information delivered from distributed content servers; a distributed content management system; dynamic content creation "wizards" to support newsgroups, training and conference registration; and multiple search capabilities, including searches over semi-structured web site material, structured searches over XML data, and hybrid searches over linked documents.

The OKC server contains only the central portal control and display code and OKC specific web content. We use Jetspeed [2] for this portal framework. Other web content is maintained on separate servers that double as content management servers. Apache-Jakarta's Slide [3] project is a WebDAV-based [4] content management system that we are evaluating.

Remote content is organized into portlets, which are mapped to HTML tables for display in browsers. The central portal server controls the arrangement and display of the portlets. Portlet content can be as simple as static web pages, or it can be a user interface to sophisticated custom services.

In the remainder of the paper we present extensions and modifications that we have made to standard portlets. We then describe two examples of sophisticated services (newsgroups and hybrid searches) that we are developing. Web interfaces for these services plug naturally into the portal container framework as portlet components.

2. Portlet Development for OKC

2.1. Introduction

The OKC Web Portal is built on an open source project from Apache's Jakarta project [5], called Jetspeed. Jetspeed provides user authentication and profiling, screen layout management and configuration, HTML aggregation from different sources, and other various portal services. We have considered Jetspeed as a starting point to implement OKC distributed content management portal. However, further research and development was needed to extend Jetspeed functionality to accomplish our goal.

2.2.Navigation Problem

Jetspeed serves as a thin Web interface with no content depth. All the content, except Jetspeed's menu and navigation links, is retrieved from remote hosts and users are directed to these hosts whenever a link is chosen. The OKC Portal presents HTML content from different sources while it keeps users navigating through the content within the portal environment. That is, we need to be able to display linked pages within the same portlet, delivering "deep" content developed by independent content developers.

Jetspeed's standard mechanism for delivering remote web content is through the *WebPagePortlet*. This portlet also uses other API functions, such as *HTMLRewriter*, to replace all the HTML links with their absolute forms.

The OKC development team has developed a new version of *WebPagePortlet,* called *OKCWebPagePortlet*. In the new version, we have assigned a unique identification to each portlet to distinguish between them. This allows us to maintain separate state in several different navigable portlets. This ID system may be superseded by future standard Jetspeed conventions.

We have also rewritten HTML links fetched from the remote content so that instead of spawning a new browser window with an absolute URL address, we now redirect this absolute URL as a parameter to our modified *WebPagePortlet*. This is summarized in Figure 1.

```
A link on a page at http://remote.host/parent_directory:

   <a href="directory/file.html">link info</a>

After WebPagePortlet processes the link:

   <a href="http://remote.host/parent_directory/directory/file.html">link info</a>

After OKCWebPagePortlet processes the link:

   <a href="http://jetspeed.host/jetspeed/portal?unique_id=
       http://remote.host/parent_directory/directory/file.html">link info</a>
```

Figure 1: How links change after being processed.

Jetspeed runs on a Turbine [6] servlet which constructs a special object, called rundata, that maintains all servlet request, session and user profiler data, and distributes it to each portlet. Following the user click, the portal request URL is formed as "`http://…/portal?unique_id=http://…/&…,`" and, from rundata objects, the URL parameters are retrieved by each portlet using their unique IDs. Finally, the new URL is used to fetch the selected page, an action which simulates Web navigation.

In case of multiple portlets used by the same user, each *OKCWebPagePortlet* stores the last clicked URL in the servlet session. When the portal view is reconstructed without clicking a link, i.e. returning from layout customizer, all the portlets retrieve their current URLs from the servlet session. If no URL is found, the initial URL from the registry is used.

2.3.HTML Aggregation Problems

All the HTML content and resources accessed by the portal have their own elements and characteristics according to the context they are produced for. The title information, metadata information, scripts, colors, font faces, styles, images, Java applets, ActiveX controls and many other technologies are widely used. However, forming a new portal page from different HTML sources requires sacrifice of features which do not work with others quite well, or scripts which do not fit into the portal environment.

OKC Web Portal discriminates among the HTML features as needed. Javascript sections are preserved but their behaviors may change. OKC includes a script library to resolve problems that content developers might face for adapting their scripts. All the HTML header tags except META tags are preserved. LINK tags are moved to the portal's header section so that developers' style may take effect. Any HTML content with frames is displayed in a new browser window outside the portal environment. Content files with systematic names for each frame can be displayed within the portal. HTML links with window and frame targets are altered to suit OKC Portal environment.

2.4. Restricted Hosts

As we solved the navigation problem, we also had to prevent HTML content from different sources displayed within the portal because of the problems mentioned in Section 2.3. We define each content host and specific location within that host as an *OKC area*. An OKC area consists of remote content in a single directory that has all been produced following OKC content guidelines. All the HTML links from an area are local and relative paths. Such content is displayed in an area portlet inside the portal. Links to other areas are valid and area portlets can switch from area to area. Any link to resources outside an area is not guaranteed to possess valid, OKC-compatible web content, so it is displayed outside of the portal view in a new window.

Each area is registered in an area configuration file in XML with its name, host and path information. Also, each area is expected to have a *stem links* file at their top level content directory. The stem links file is an XML file that defines the content's top level menu. This file contains a menu title, description, and a list of menu items, each having a relative path from the top level directory to the content file, an item title and a brief description. The OKC Web Portal provides area sensitive menus along with the area portlets which constructs a sub menu while users navigate through the content. Any changes in areas are detected and the sub menu is replaced with another menu using stem links files inside the content.

3. XML Messaging for News Groups

The OKC Web Portal is enriched by collaborative tools. As one of the first attempts to make this happen, we have added newsgroups capability to the portal where users can interact through both online and offline messaging. The OKC not only provides interfaces to read and

post messages but also allow e-mail distributions to and from group subscribers. This service resembles mailing lists; however, the architecture is built on distributed modules operating separately and communicating by a JMS-compliant event brokering system [7,8] so that each module can serve as a Web service to multiple clients or other services as systems grow and have new functionalities. This represents an example of using forms with XML targets.

The newsgroup system's major components and interactions are shown in Figure 3.

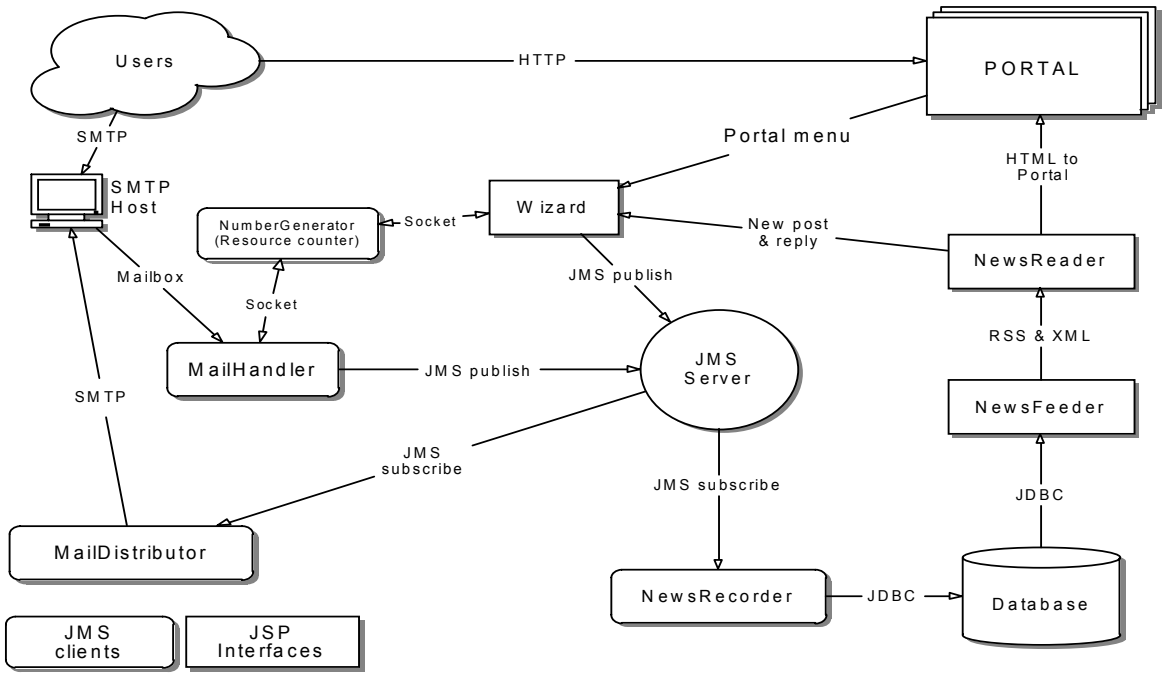

Figure 2 OKC's newsgroup architecture supports both email and JSP clients and readers.

Newsgroup messages are nothing but events for the system, whose structures are defined by an XML schema, enveloped in SOAP [9] messages with attachments, and handled as other system events are handled. Newsgroup modules listen to distributed events and distinguish the newsgroup messages from others and act accordingly.

There are two major event generators in newsgroups architecture. These are the *News Wizard* and the *E-mail Handler*. The Wizard is a JSP[10] interface based on the event schema. For newsgroup users, it is an interface to post messages to the system. After users post their messages, the Wizard generates XML event instances and publishes them to the event broker as system events. The second major event generator is the Handler. It interacts with an SMTP host, validates and converts users' e-mail messages to system events.

Each event is identified by a unique URI such as gxos://okc/events/2334 and messages contained by events as gxos://okc/newsgroups/agroup/44332. Each URI corresponds to an XML document or XML metaobject whose schema and management are defined by GXOS [11]. Each XML metaobject contains information about where and how an actual resource is accessed and preserved.

The *Number Generator* service is deployed to ensure a system-wide unique naming. This service is used by event publishers before any URI is assigned to events and messages.

As events are published, there are two subscribers of such events, the *News Recorder* which provides persistency for system events. Each user message is recorded in a database for later referrals. The other subscriber is the *E-mail Distributor*. The Distributor detects events with user messages and sends them out to the subscribers of the news group the message was sent to.

Recorded events or user messages can be reviewed as a list of resources in Rich Site Summary format [12], which is meant to publish recent content changes of a Web site in an XML format to remote subscribers. RSS contains a topic, description, and items with link titles and addresses to the related resources. The *News Feeder* module provides such RSS feeds to any other parts of the system. Any resource chosen from the items list can be retrieved from the database by using its address information through the Feeder.

The front-end of the newsgroups architecture is the *News Reader*. This module interacts with users and retrieves RSS feeds for selected groups or news messages in XML if a particular message is selected to display. The Reader uses XSLT [13] transformers to generate HTML content for the portal from XML messages.

4. Hybrid Search Prototype

We made a test prototype of the hybrid search for the XML documents attached with external files. An XML document can have link tags, which designate external documents. The external documents may be Microsoft Word files, Microsoft Power Point files, PDF documents, or Post Script files. Hybrid searches allow us to simultaneously search semi-structured documents and structured metadata about those documents. For this, we will incorporate Oracle database and searching tools [14,15,16,17].

Actually, we cannot know specific information of the paper document without an XML instance that represents meta-data with a title, authors, affiliations, the source, and the published year. The benefits of the meta-data are not only the particular meta-information but also the performance improvement with narrowing the group to be searched for the context index. However, the large amount of XML documents may drop the performance for extracting particular nodes from XMLType columns in the Oracle database tables. In that case, we can consider ordinary indexed column types mapping to the nodes of the XML instances. In the test prototype, an XML instance has a single external document to simplify the architecture. We made an XMLType column type table for the XML instances, a BFILE large object type column table for the document, and a relationship table described the relationship between the XML and the file. The Entity-Relationship diagram for the test prototype is in Figure 3.

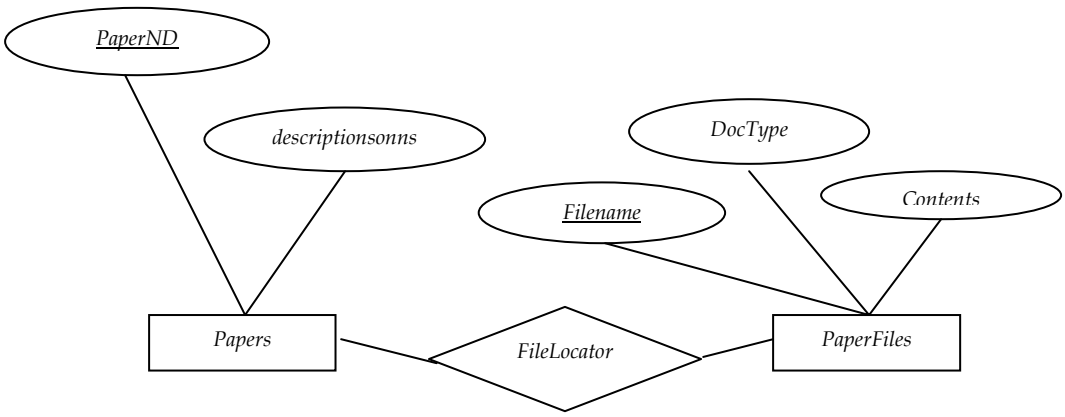

Figure 3 E-R diagram of the test prototype

The *descriptions* attribute of the *Papers* entity set represents the XML instances with XMLType object type and the *PaperND* (Paper Name and Directory) is a key field with a unique value of Name and Directory in the *Papers* table. The *Contents* attribute of the *PaperFiles* entity set has the BFILE large object type and the actual file will be stored in the designated subdirectory in the outside of the database. The *DocType* column has a filtering option: *BINARY* will be filtered and *TEXT* is not necessary to filter when indexed. The additional update, insert, or delete will

change the status of the index and the new information will be reflected into the index using *synchronize* package included in the Oracle 9i database system. The *FileLocator* table shows the relationship between two data tables: *Papers* and *PaperFiles*.

Users interact with the search tool through a web interface. User queries are handled by a Java servlet that makes a SQL query from the parameters passing from the user search Web page, gets the result set querying through JDBC connection from the database tables.

5. Summary and Future Work

In this paper we have described extensions to portlet capabilities that are needed to support entire remote areas within a single portlet. We have also described important example services: the newsgroup system will serve as a prototype for all of our dynamic content creation and management, and hybrid search capabilities can be applied to search online libraries. Important future work includes more sophisticated content management services, including page validity services that check HTML compliance and message-based authoring systems similar to the newsgroup system. We also plan to develop portlets that support multimedia content. Finally, we must also support WSRP-style web services for finding and binding to distributed portlets. This will require support for distributed events within a web services framework, which will be an important area for development.

6. Acknowledgements

The Online Knowledge Center is funded by the US Department of Defense's High Performance Computing Modernization Program through the Programming Environment and Training initiative. We gratefully acknowledge their support.

7. Reference

[1] Web Services for Remote Portals: http://www.oasis-open.org/committees/wsrp/
[2]Jetspeed Overview: http://jakarta.apache.org/jetspeed/site/index.html
[2] The Jakarta Slide Project: http://jakarta.apache.org/slide
[3] WebDav Resources: http://www.webdav.org
[4] The Apache Jakarta Project home page: http://jakarta.apache.org
[5] Jakarta Turbine: http://jakarta.apache.org/turbine
[6] Java Message Service API 1.0.2: http://java.sun.com/products/jms
[7] Narada Event Brokering System, http://grids.ucs.indiana.edu/ptliupages/projects/narada/
[8] Simple Object Access Protocol (SOAP) 1.1: http://www.w3c.org/TR/SOAP
[9] JavaServer Pages Technology: http://java.sun.com/products/jsp
[10] Garnet XML Object Specification: http://aspen.ucs.indiana.edu/project/gxos/
[11] RDF Site Summary Specification:
http://groups.yahoo.com/group/rssdev/files/specification.html.
[12] Extensible Stylesheet Language Tranformations: http://www.w3c.org/Style/XSL
[13] Oracle Corporation, Oracle 9i Application Developer's Guide – XML, June 2001.
[14] Oracle Corporation, Oracle Text Application Developer's Guide Release 9.0.1, June 2001.

[15] Oracle Corporation, Oracle 9i New Features Summary, Technical white paper.
http://www.orcle.com/xml/documents, October 2000.

[16] Oracle Corporation, Oracle Ultra Search Architecture, Technical white paper,
http://otn.oracle.com/docs/products/ultrasearch/, May 2001.