Web Portal Solution is better than having only Distributed Applications for Organizations using Intranets

Master's Thesis

Student Number: 191131

Name: Saif ur-Rehman Butt

Dated: 9 December 2006

Submitted to: Dr. M.R. Mousighi

Dedicated to my wife who stood by me in thick and thin and always believed in me and encouraged me to accomplish my goals,
Thank You Love.

Table of Contents

Chapter 1: Introduction 1
Statement of the problem 2
Purpose of the study 4
The web portal 6
The Portal Solution 7
A web portal strategy 15

Chapter 2: Web Portal Features 22
The web portal advantages 23
Web portal types 26
Web portal software functions 36
Web portal universal features 41
Web portal user interface 46

Chapter 3: Web Portal Strategy 49
Importance of web portal strategy 50
The web portal strategy 51
Web portal concepts and technologies 53
The IT environment 55
Building the web portal business case 56
Information technology analysis and results 58

Chapter 4: Portal Application Design and Development Guidelines 61
Introduction 62
Websphere portal 62
Introduction to object oriented design patterns 63
Introduction to portlet framework 69
General approach to portlet development 71
State pattern portlet framework 72

Chapter 5: Portlet Development Guidelines 95
The Portlets 96
Pattern packaging guidelines 102
General session management guidelines 104
Performance considerations / analysis 112
Performance guidelines 118

Chapter 6: Portalization Approach for Existing Applications 131
Application Integration 132
Considerations 138

Chapter 7: Summary and Conclusion 140
Summary 141
Conclusion 144

Chapter 1: Introduction

STATEMENT OF THE PROBLEM

During the past several years nearly every new application and idea created by the software industry has made the jobs of employees more complex and difficult, rather than simplifying their responsibilities. Often the users have to access several applications to do their assigned tasks. For example, enterprise resource planning (ERP) system allows the user to review inventory and backlog information. However, for checking inventory status of a particular part, it is necessary to open the ERP system and navigate through numerous screens. Then a second application or system has to be launched to enter the purchase order. Any problems or issues that may be encountered may require access to customer information that is housed in the customer resource management (CRM) system. Thus employees or users responsible for making strategic decisions have even more difficulty trying to do their jobs effectively.

Many Organizations use custom made distributed applications running on the intranet to full fill their local needs. These applications are either desktop applications or web applications. These applications are made up of distinct components running in separate runtime environments, usually on different platforms connected via a network. Typical distributed applications are two-tier (client-server), three-tier (client-middleware-server), and multi-tier (client-multiple middleware-multiple servers). In spite of having these custom made solutions, there are still a huge number of multiple information files being used and transferred among different departments. These files are in the shape of Word

documents, Excel spreadsheets, Power Point presentations, Pictures etc. scattered over the Intranet owned by different departments or users. These files or scattered information can be centralized under one umbrella, for that we need a better content management solution. In many cases the data sources, systems, and applications located throughout the organization need to be combined to present the summarized information or desired report format that executives expect to review. The IT department likely has several initiatives and activities associated with taking the information available from transactional systems and converting it to a format needed to maintain an enterprise information system (EIS) or decision support system (DSS), or the systems that provide the summarized information this group of employees needs to make strategic decisions. To find an answer to a question, make a decision, or take some action, employees spend an enormous amount of time traversing the maze of these disconnected applications and systems.

This is only part of the problem. Corporate wide systems are complex and designed for a specific purpose and function, so the IT department is required to deploy many different and often unrelated applications and modules to fill the information and processing needs of the entire organization. In addition, an incredible amount of training time is needed for an employee to learn how to effectively use such a complex suite of applications and all of the processes and steps involved to complete their assigned responsibilities. Only a small fraction of the organization's employees know how to use all of the functions of one system, much less all of the systems and databases that affect their job.

PURPOSE OF THE STUDY

Over the last 4 to 5 years, web portals have risen in popularity as a way of aggregating, organizing and presenting content in a highly uniform, customizable and personalized way. As the technologies that enable the creation and management of these web portals have evolved, it is not only information content that is being offered, but application functionality as well. With application functionality making its way to the web portal, a whole new dilemma is encountered as developers attempt to adapt their application functionality to the characteristics and behavior of the web portal.

The corporate intranet was originally designed and implemented to meet this need for shared information across the organization. Using the corporate intranet, employees are able to access corporate information using a web browser such as Microsoft Internet Explorer or Netscape Communicator to find forms, open applications to perform their jobs, and review a customer's project status, and for many other activities. The corporate intranet solution provides navigation to different enterprise systems and documents.

Corporate intranets everywhere are responsible for hosting a multitude of applications and an exponential growing number of documents. These applications, documents, related tools, and enterprise systems need to be made readily available for employees to use. This is a difficult task for IT departments, and it has created information access, knowledge-sharing, and security problems. As intranet sites grow larger, a new set of problems has been created.

The purpose of this study is two-fold; one is to build a strategy to improve knowledge sharing and decisions made about information access throughout the organization because the organization needs to share information among employees, locate information that is difficult to find, push information to users, or create a central location to navigate through data that employees can benefit from and show the usefulness of proper content management or information management by suggesting a proper solution and second is to look at from the developers perspective that how the changes can be made to put old applications work under portalization and the tips to use in developing new portal-ready applications. WebSphere portal development platform will be used to explain the whole procedure of portal application development and deployment.

Therefore, this thesis is divided in two parts. Part I (from chapter 1 to 3) will focus on designing web portal strategy and suggesting a solution. Part II (chapter 4 to 6) will focus on providing tips and guidelines to enable the developer to create portal enabled applications that fully exploit the features of today's portal infrastructures and will prove that how the portal solution is better than having only one or even more web applications.

THE WEB PORTAL

A portal solution is a facilitator to knowledge that helps employees to make timely decisions. There are several issues and perspectives to consider when looking at how the organization can leverage knowledge for the purpose of making better, faster, consistent, and more informed decisions. The perspectives are operational and strategic. Operational issues are the ones faced by employees responsible for completing transactional, day-to-day, and well-established tasks. Employees responsible for operational issues in the organization are constantly looking for ways to improve or simplify existing processes or tasks. Strategic issues are the ones faced by employees responsible for ensuring that the overall mission of the organization is met or exceeded. Employees responsible for making strategic decisions in the organization are tracking financial and other information that accurately measures how well the organization is performing. The primary objective for a portal software solution, then, is to create a working environment that users can easily navigate in order to find the information they specifically need to quickly perform their operational or strategic functions and make decisions.

The portal solution is a collection of technologies—software and infrastructure—that work together to aggregate a selected subset of information into a central location. From there, different employees can then easily access information that's relevant to their roles or business and personal requirements and in the process work more effectively with each other. A web portal is usually structured around roles that are found inside the organization.

One of our first tasks to complete when considering a portal strategy is to establish a portal definition for the organization. We want to establish a standard definition along with a collection of portal objectives specific to the organization. We can then use this definition and list of objectives to evaluate the software features available from portal vendors and other sources to find the best portal solution available for the organization.

THE PORTAL SOLUTION

There is an intranet concept that resolves these problems and issues the organization is currently faced with. The concept is to create a web portal. A portal is an intranet "window" that presents information to users and an intranet "door" that allows users to pass through to reach selected destinations. The web portal creates the central location where navigation services are available for employees to find information, launch applications, interact with data, identify collaborators, share knowledge, and make decisions. The definition of a web portal therefore is:

A browser-based application that allows knowledge workers to gain access to, collaborate with, make decisions, and take action on a wide variety of business-related information regardless of the employee's virtual location or departmental affiliations, the location of the information, or the format in which the information is stored.

Web portals can provide a successful intranet strategy for the organization. Web portals have several features that are used to implement knowledge management initiatives in the organization. These include:

A consistent view of the organization

Employees must identify and interact with many different data sources. The web portal can provide access to an index of report catalogs and direct users to the right information source. The type of information the employees need to access might be structured (e.g., data marts, data-warehouses), unstructured (e.g., e-mail messages, word-processing documents), transactional (e.g., creating a customer order), or collaborative (e.g., scheduling a meeting, commenting on the department budget). The web portal is built on existing content repositories, applications, and data marts that users can immediately access without having to know the actual location of the data or which application is used to read or update the data. By creating an interactive environment with every aspect of the organization and every type of job-related activity represented, the result will be a better forum for making decisions, sharing best practices, and exploring new ideas.

Information organization and search capabilities

The web portal can provide a format to implement a common taxonomy and consistent semantics for the organization. To facilitate employee access to corporate information in a structured fashion, a web portal menu hierarchy is required so that employees can understand and quickly

navigate the information. This navigation system creates a common naming convention for processes, procedures, and activities in the organization. This menu hierarchy also serves as a logical breakout of how information is "filed" in the organization and which part of the organization is responsible for keeping the information updated.

The web portal creates a central location for users to find and access corporate content. The portal is not designed to host information or to be a content store. The content presented in the web portal should continue to reside on the file server or in the host application that controls the data storage and business rules of the information or report that is displayed in the portal. Implementing a knowledge management solution that maintains the existing status quo for data storage and business rules and processes is critical to sustaining the integrity of the information presented in the web portal. The web portal solution creates a wrapper of services around the information that is hosted in the user interface. These services include user assistance, search facilities, identification of relevant information (e.g., related white papers, additional information about the customer, how other departments use the report or information being reviewed, who to contact if one has questions), facilities to create discussions and interact with a community of users, and the ability to drill down into more detail and add value as a navigation aid.

It is designed to provide a knowledge desktop with a user interface that displays or presents information in a browser client from data sources that reside throughout the organization. There are hundreds of ways that the organization

affects relationship with employees. Employees should be able to construct queries, search for correlation in the data, plot the data, and move the results into other applications such as spreadsheets and word processing documents. This flexible access to data and the tools necessary to manipulate, navigate, and locate the data available to them is critical.

Direct access to corporate resources

This functionality gives employees and business professionals the means to communicate and collaborate with each other. To share knowledge throughout the organization, employees need to publish and distribute information that can be referenced and accessed from a central location. By providing an environment to share information, employees should be able to collaborate with each other and there by improve processes.

We also should establish communities or team rooms where employees regularly share information and ideas, examine and weigh options, act upon the organization's values and beliefs, and learn to improve the quality of their decisions. Communities provide the best context and most promise for addressing this challenge. The goal of incorporating communities in the web portal solution is to teach employees how to make choices that will sustain the organization over the long term. Establishing a growing sustainable community's movement that supports continuous interaction as part of the web portal solution creates an unprecedented opportunity for employees to learn from one another.

Employees need to communicate, collaborate, and coordinate with other employees and systems in the organization. These features can be implemented

through the corporate messaging system, discussion forums, automated alerts and notifications, or the automation of approval processes and associated status information. All of these features are available from corporate applications and systems that are made available from the web portal solution. How each of these features is implemented into the web portal solution will differ, depending on the portal software selected by the organization. Some features can be implemented by simply configuring the portal software, while other features require custom development of the portal software development environment.

Direct links to reports, analysis, and queries

Applications are built to support and maintain multiple groups of individuals throughout the organization. Corporate wide systems are complex and designed for a specific purpose and function, so organizations have to deploy many different applications and modules to fill the information and processing needs of the entire organization. The web portal directs users to locate and launch the right module of an application to do their job or complete a specific responsibility with the right security. This allows employees to be directed to only the components or screens of an application that are specific to their job. As a consequence, the web portal solution has an additional benefit in helping to reduce training costs. Because the web portal removes the complexity of understanding the entire application functionality, users can concentrate on only the concepts and steps essential to completing their assigned responsibilities. Although employees may still require training classes and

support from more experienced users to become proficient at using a complex suite of applications, training can be concentrated on the systems, applications, and data sources that directly impact their job. The training and cross-training challenge continues to expand as new employees, processes, and systems are integrated into the organization. By pushing selected information into a web portal solution, training requirements may be reduced and focused specifically to individual roles in the organization. From within the web portal solution, employees should be able to search data sources, look for patterns in the data, create graphs and presentations from the data available, and move the results into other applications or other formats used to share new information throughout the organization. This flexible access to data and the tools necessary to manipulate, navigate, and evaluate data creates an environment that promotes enhancements and improvements to existing process and services.

Direct links to relative data

Employees are given an awareness of relevant information that is outside the immediate domain or department of their primary job requirements. Such information is generally presented as links that can be easily located on the web portal context pages. The purpose is to help employees make speedy decisions while doing their job without experiencing information overload. There may be business-essential information that is not directly a part of the decisions an individual employee makes. (e.g., related white

papers, current industry information that might be helpful, customer account information, customer order information, how other departments use the information or report, where to find user assistance about the current process being followed, who to contact if they have questions). The information is nonetheless relevant to understanding how the employee's current activities affect other employees in the organization and how the activities of other employees directly or indirectly affect that person. This capability facilitates fast action, creative problem solving, and a better understanding of the entire company. To establish a portal interface that provides seamless integration between data and employees in the organization, we must define and maintain several identity profiles or reference maps. These profiles and reference maps work together and cross-reference each other to present information and hyperlinks in the portal interface. The user thus has a personalized desktop that opens in a browser with the organization's corporate information, personal mail and calendaring information, and job-specific information. All of the information available in the web portal interface can become the central focus of the context window with a few simple clicks of the mouse. All the hyperlinks and drill-down features associated with the data in the content window should be context sensitive. Using the identity profiles and reference maps to define these relationships, this type of access to relevant information can be made easily available in the web portal. The job of the IT

department is to define, create, and maintain the data source that controls these identity profiles and reference maps.

Individual identity and personalized access

The web portal facilitates the important role of presenting the right and requested information to the user based on the access and security defined in the user profile. Employees are defined by their digital business identities that describe their roles, activities, skills, and positions in the organizational chart. Workflow processes, job-related functions, and corporate functions can be facilitated from the intranet through the web portal. The information presented to each user in the web portal also has certain useful characteristics: First, it can be context-sensitive based on the employees' role and responsibilities in the organization. Second, it can be personalized by each user to arrange and organize the information in the web portal to that individual's specifications through configuration. There needs to be a balance between the information that must be constant in the web portal and the information or features that can be defined or configured by each portal user. The main portal menu hierarchy is an example of a piece of information that should remain constant for all portal users. Examples of information that should be configurable include adding favorites or bookmarks inside the web portal so a user can easily find information that is referred to frequently. Users should be able to configure a personal tabbed page from each menu option or some other

feature that allows content pages to be organized and reorganized as required by the portal user.

A WEB PORTAL STRATEGY

The web portal is an excellent format to share information about each role or responsibility, process or activity, and department or workgroup throughout the organization. The process owners or business domain experts for each of the areas of knowledge in the organization can be designated to establish and maintain the information available for employees to access through the portal desktop. Whether they use business domain experts extensively or only to monitor documents and discussions that they have knowledge about, they need to be responsible for some of the content presented through the web portal. Employees should have access to information from all the areas of expertise needed to perform their assigned responsibilities. All employees in the organization should be required to access documentation and information about the organization. The web portal solution presents data available from enterprise systems and other information published by business domain experts together in the portal desktop; as such, it serves as a single point of access for users.

Our goal is to determine what published information is required by the proposed web portal system, identify the business domain experts, and determine how the IT department and these business domain experts can make information accessible. The implemented result should allow employees to get the latest information they need about the role, process, or department without having to search through several systems or make several phone calls. To accomplish this

result we need to consider third-party applications and web portal software products. Creating the portal desktop required achieving the defined objectives means establishing the correct combination of information and services required by the business domain experts, web-based services and technologies, and software applications.

The organization needs to consider and build the web portal solution in stages. There are several universal features that are incorporated or built into a web portal solution. These universal features include:

- Self-service applications (i.e., access to discrete enterprise data)
- Web portal (i.e., aggregation of self-service applications)
- Comfort and familiarity (i.e., user personalization)
- Understanding and knowledge (i.e., navigation personalization)
- System intelligence (i.e., behavior assistance personalization)

The first generations of web portal solutions are self-service applications. Each self-service application is developed and added into a central knowledge desktop. These self-service applications collectively become the web portal. In many cases the personalization requirements of the web portal need to be globally or universally defined and implemented when the first self-service application is added. The personalization requirements can be enhanced and updated as the web portal evolves and grows.

Successful completion of the first self-service application makes the intranet content included in the web portal accessible to all employees complete with search, browse, navigation, and personalization services. The second-generation

implementation compiles or organizes several self-service applications into a collection of central corporate "windows" or portals. In effect, we are extending the self-service applications to incorporate web portal services that include additional applications, additional data points or data sources, collaboration, and workflow processes. The third generation of the web portal solution should complete as many universal features as possible, including personalization, decision making, and knowledge management initiatives as needed. These features are defined and included in the personalization aspects of the web portal. Personalization of the self-service applications and web portal services allows a user-defined and managed configuration of the information for each user's unique web portal experience. Some personalization features to consider include:

- Identifying of knowledge experts and data sources directly and indirectly related to completing each user's assigned responsibility.
- Pushing information to users on the basis of their responsibilities or interests.
- Configuring the web portal applications and information systematically based on tracking the preferences and behaviors of web portal users.

These web portal universal features are a way to outline and measure the capabilities to be included in the web portal and to determine what personalization capabilities to include in each development and implementation phase.

A web portal solution is a web application that is going to be made available to all employees and used to service all workgroups and individual department needs. Web portals are built using infrastructure technologies (e.g., web servers, data warehouses, application servers). The organization must align the web portal investments and thin client (intranet and extranet) infrastructure commitments.

As we create the web portal strategy and business case for the organization, we should also consider the expectations of the extranet customers and partners. The extranet partners require the same knowledge experts, data warehouses, and access to applications to complete their activities and interactions with the organization. We should make certain that the web portal strategy includes directories and security plans to operate in concert with extranet systems and service requirements.

When Microsoft Corp. first began conducting usability studies in the late 1980s to figure out how to make its products easier to use, its researchers found that out of every ten users, six or eight people could not understand the user interface and could not use most of the features. Software portal products are being developed to provide the user interface and functionality requirements (e.g., navigation, search, data connectivity) of portal solutions. We must consider the user interface functionality and requirements equally as important as the back-end and security functionality and requirements of the web portal solution.

The web portal strategy team should include individuals whose combined backgrounds represent employee management skills, project management skills, business process analysis, and several years' experience with the organization.

Normally the first phase or iteration of the web portal solution includes one or many self service applications. A self-service application is a unique instance of an isolated job function, responsibility, or process that can be encapsulated as a system or subsystem that is defined as a part of the complete web portal solution. The web portal solution is deployed in several iterations or design phases in order to reevaluate universal or global features as the portal application incorporates functionality in the higher levels of personalization and navigation. This reevaluation process is important to make sure that all portal windows and self-service applications have consistent behavior and functionality throughout the web portal solution.

In addition, there are some notable benefits associated with a staged delivery of the web portal design and implementation. They include:

Plan to deploy the most critical functionality earlier.

Identify the most important business functions and processes in the organization to be added to the web portal first. Employees who need these functions can begin using the web portal application while the next function or process is being developed, tested, and deployed. Staged delivery can be a valuable approach for gaining acceptance of the web portal solution in the organization.

Reduce risks early.

The business plan can outline how a staged delivery of the web portal solution emphasizes risk management throughout the project. Delivering the web portal solution in stages reduces the technical and political risks

of unsuccessful integration of software, technology, and user acceptance. Management risks are reduced through frequent deliverables that demonstrate progress to everyone watching the project. Building opportunities to revise plans into each stage of the delivery further reduces planning risks.

Identify problems earlier.

The more frequently new web portal functions are added into the web portal solution, the more frequently feedback is collected and forwarded to the planning team and the developers of the system from the employees using the system.

To make more options available sooner.

The planning team and the development team can consider addressing changes requested by the employees using the web portal solution. The staged delivery approach allows the planning team and the development team to guarantee that changes will be considered periodically by building opportunities for improvements into new delivery stages.

Specific business and technological advantages of a web portal solution include the following:

Business Advantages

- Planning and implementation at the roles and responsibilities, activities and processes, or department and workgroup level ensures a manageable deployment.

- Ownership at the roles and responsibilities, activities and processes, or department and workgroup level allows executives and managers to plan a web portal tailored to the unique needs of employees throughout the organization.

Technological Advantages

- Rollout provides a degree of flexibility in the implementation of functionality and connections between web portal services and data sources.
- Scalability lends itself to supporting a large variety of features, functions, corporate standards, and connected systems. This allows the web portal to provide a number of services (e.g., user assistance, search capabilities, discussion forums) that grow in complexity with the addition of self-service applications (e.g., sales force automation, customer relationship management) and enterprise systems (e.g., Oracle Financials, Microsoft Office applications).
- Growth allows the web portal to evolve as the requirements and available technologies change.

Chapter 2: Web Portal Features

THE WEB PORTAL ADVANTAGES

The organization can gain several advantages by implementing a web portal solution. These benefits include (among others):

Better decision-making capabilities: The structure and content available from the web portal desktop should provide the appropriate detail and direct access to information in the organization for employees to improve and enhance decisions they make while doing their jobs and completing their assigned responsibilities.

Improved semantics: The web portal solution provides the framework to create a consistent view of the organization through the use of consistent terminology and navigation hierarchies.

Improved information organization and search capabilities: The portal solution focuses on the layout of individual desktop screens that are organized around the way employees work rather than around the applications they use. An additional feature is to provide search features to locate documents and information available inside the organization or on the World Wide Web.

Direct access to knowledge and resources: From a central portal desktop, employees can focus on the aspects of applications, data sources, and the experience of other knowledge workers to complete their assigned responsibilities.

Direct links to reports, analysis, and queries: Summarized or status information that needs to be tracked, published, and presented is incorporated into the portal desktop. These reports provide several services that include (1) information to

educate employees and (2) methods for evaluating and measuring the success of the organization's performance objectives and defined metrics.

Direct links to relative data points: This benefit encapsulates the ability to traverse content inside the portal desktop that is relevant to the primary responsibilities and interest of the employee.

Personalized access to content: Personalization provides the ability and functionality required for each employee to organize, define, and rearrange the portal desktop to work effectively and efficiently.

These benefits are major elements or software functions that are built into the portal desktop used by the employees. These software functions work together in different combinations to achieve the portal benefits identified as important by the organization. Employees interact with the portal desktop, which allows the organization to address the corporate objectives identified to improve the entire decision cycle of employees throughout the organization. This decision cycle consists of three steps:

- Gathering information.
- Engaging in research and collaboration activities.
- Taking action.

The web portal must provide features and content for each step in the decision cycle. The portal is a facilitator designed to draw the employees into a common community. It is a tool to exchange and improve relationship with the employees and the customers. The goal of the portal desktop is to implement self-service, self-help, and self-discovery in aspects of our business where it makes sense.

For example, employees that are experts using a specific application or spend most of their day using many aspects of a specific system to complete their assigned responsibilities will continue using these applications and systems. The organization may decide that the portal desktop can provide two separate and distinct benefits to these knowledge workers. First, it provides access to information, applications, and interactions in the organization that they are often not aware of or familiar with. Second, it provides a format for them to publish their expertise, be recognized as experts, and be actively involved in improving processes directly or indirectly related to their responsibilities.

The concepts, advantages, and features of a web portal solution for the organization are undoubtedly worth consideration. The following issues are worth the consideration:

- The types of portals available from portal vendors, including their basic features and services
- The functions to include or incorporate as software requirements for our web portal solution
- The universal features the organization needs to establish as standards for our web portal solution
- A definition of the components in the user interface that represent the web portal software functions and universal features

WEB PORTAL TYPES

Portal software vendors offer a collection of several features and functions that work together to provide the benefits the organization can get from a portal solution. A wide variety of products and vendors in the business intelligence, enterprise resource planning (ERP), document management, search engine, and other markets partially fill the definition of a "corporate" portal. A review of the portal market segment identifies at least nine different types of web-based applications that have labeled themselves portal solutions. A collection of these portal market segments must be combined to match our definition of a complete and scalable web portal solution. The relationship of these nine web portal segments and the portal terminology used in our web portal definition is diagrammed in **Figure 2-1**.

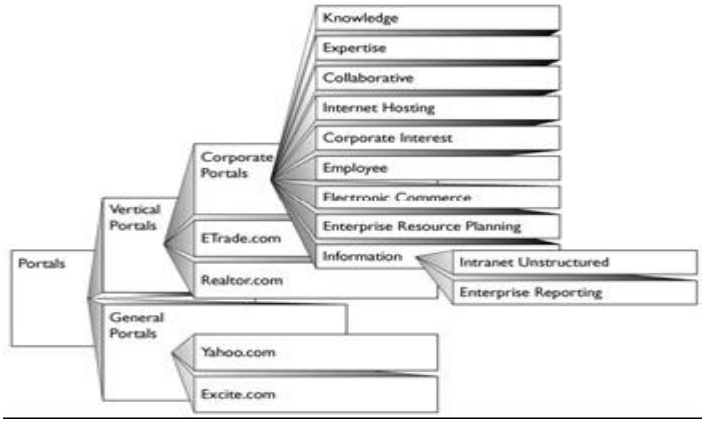

Figure 2-1: Web portal market segments.

Information Portal

This is the most general category and includes vendors that emphasize connecting to many classes of data sources. The two types of information portal vendors are:

Intranet unstructured: These vendors provide systems that automate the searching, categorizing, organizing, and publishing of intranet-based information. Examples of vendors in this market segment include Verity and InfoSeek. These systems are easier to install than enterprise reporting systems and have features similar to public or general Internet portals such as Yahoo! and Excite.

Enterprise reporting: These vendors have strong query and reporting applications. Most vendors in this category have come from the data warehousing or business intelligence space. Business intelligence vendors in this market segment have announced a subset of features found in their online analytical processing (OLAP) tools to provide data mining for the masses. Market segment vendors include Oracle and Brio.

Enterprise Resource Planning (ERP) Portal

This category primarily consists of browser front-ends to the vendor's own ERP system. Emphasis is on providing information from the ERP system to the casual user, using a web infrastructure. Every ERP vendor has announced a product in this category, including:

- SAP
- Oracle

- PeopleSoft
- Baan
- J. D. Edwards

There is some cross-functionality between portals in this category and e-commerce portals owing to the tight integration needed between back-office systems and front ends of e-commerce customers.

Products in this portal category primarily connect to the ERP system for which they were developed. Thus, to take advantage of an ERP portal, the organization must have an installation of the ERP system. The ERP system is the data source that is presented in the ERP portal solution. Employees gain access to the enterprise wide capabilities of the ERP system while taking advantage of a web-based work environment. These ERP portal solutions provide the following features that allow individual employees to customize their ERP portal solution:

Personalization: The workspace can be personalized for individual preferences and corporate roles that are applicable to each user.

Single point-of-access: There is a website used to access screens, reports, and features of the ERP system. Included in this website is the ability to add features that create links to launch or include aspects of third-party applications.

Administration, maintenance, and security: There are administration and maintenance features to manage access and permissions to ERP data and business processes. Security features that are incorporated in the ERP system are used to control the web-based work environment with encryption and

additional security requirements that are identified by the organization's information technology (IT) standards.

Electronic Commerce Portal

Electronic commerce (e-commerce) portals are aimed at customers for the sole purpose of online retail or procurement applications. These solutions create an Internet-based business relationship between the organization, customers, partners, and vendors. E-commerce portals use web browsers to create process and action, networked supply chains, and customer relationships and to establish new market activities. Organizations concentrate on several goals and objectives for their e-commerce portals. One feature is to be able to tailor offerings through personal profiles. Amazon.com concentrates on streamlining all of the business processes that affect customers to create a comprehensive set of retail services. The objective of the Dell Computer's e-commerce site is to let customers help themselves through the entire sales experience including pre-sales marketing through order entry and post-sales service and support.

Employee Portal

This category has a definite human resources focus. Employee portals combine ERP and human resource information system (HRIS) access with the addition of work scenarios or scripts that lead users through the processes and routines of predictable daily work, rather than requiring users to piece it together themselves. The goal of employee portal products is to promote a high level web-based work environment with assistance or help that assembles structured,

unchanging processes into role-based routines. Services and functionality available for employees in the organization to take advantage of include:

Human resources functions

- Managing positions and associated compensation.
- Recruiting, hiring, and training employees.
- Promoting, relocating, and retiring employees.
- Incorporating local and international regulatory requirements.

Benefits administration

- Pension administration
- Payroll administration
- Stock administration

Corporate Interest Portal

This category consists of corporate line-of-business portals available from the Internet. The corporate interest portal provides listings of business-to-business resources, services, and business products that might be of interest to the organization. The information available directs users to online and offline sources. These Internet sites use search and directory hierarchies to direct users to the most relevant information they need without having to search the entire World Wide Web for the service or resource they are trying to locate. These corporate interest sites are often categorized into identified industry verticals that contain dedicated subject matter specifically focused to a line-of-business or specific business activity.

Internet Hosting Portal

Internet hosting portals provide line-of-business-oriented application hosting in which the vendor runs the system on its infrastructure. Services provided by Internet hosting vendors that can be made available to the organization include:

Download services: These services are used for distributing files (including software), white papers, and other material. This service saves an organization costs on printing, order entry, fulfillment, and shipping.

Application hosting: This service is typically for hosting Active Server Pages that are created and maintained by the organization and made available through the Internet hosting provider.

Internet connectivity: This service provides everyone in the organization access to the World Wide Web.

Security services: These services guarantee that downloads and mail messages and other Internet services are secure transactions so that the organization's data cannot be accessed or seen by any unauthorized users.

Even if the organization's website is hosted outside our internal network by a portal vendor, the organization should consider a unique domain name. The domain provides the organization with:

- A unique identity on the Internet for the organization to use
- A unique web address (e.g., www.mycompany.com)
- The ability to route mail to the domain (i.e., web address)

Collaborative Portal

Collaborative portals are applications that focus on communication features such as team rooms, project management tools, discussions, chat rooms, and e-mail. These Internet sites and related applications or products are designed to support three key activities:

Communication: This means creating an environment where team members can communicate and share information regardless of time zones and geographical boundaries.

Collaboration: E-mail, electronic document storage and retrieval, corporate discussion forums and information distribution, and personal time management resources are included in this category.

Coordination: Tools for creating and documenting processes that are shared with the whole organization provide the capability to analyze and simulate process and action or workflow. Business process modeling allows the organization to update workflow activities with the best possible solution being implemented.

Expertise Portal

Expertise portals provide informed advice, decisions, or recommended solutions for a given situation or submitted question. These solutions are designed to incorporate knowledge and reasoning methods that are not easily represented in traditional computing approaches. Expertise portals extend the boundary of what can be automated to include tasks that require reasoning, symbolic processing, and problem solving. The goals of the expertise portal are to be accessed by

many people, to represent knowledge that may reside with many individuals, to preprocess information to increase the availability of expert information, and to gain access to expertise immediately. The expertise portal fills a number of functions:

Librarian: Helping employees find, organize, and interpret information required to carry out a task.

Advisor: Sharing the specialized expertise needed.

Instructor: Helping employees learn a task.

General assistant: Taking care of defined routine tasks

The expertise portal is composed of two somewhat independent parts: an inference engine and a knowledge base. The inference engine is the control structure used to retrieve and submit requests to the knowledge base. The knowledge base is where the real power of the expertise portal exists. This is the coded pool of rules, insights, and knowledge that are available for employees to access. This structure provides two important features for the expertise portal. First, the expertise portal allows the knowledge base to be modified, updated, and expanded. This makes it easier to keep the information available current with changes in the industry, or with changes in user requirements. Second, an explanation of the reasoning behind a conclusion can be allowed. This feature is required to provide the credibility and confidence that employees will require before accepting the information provided into use.

Knowledge Portal

A knowledge portal solution that completes our definition is a combination of all the aforementioned market segments with a concentration on the following capabilities:

- Retrieving core information from corporate IT systems and presenting it according to the preferences, roles, and specific tasks of individual users.
- Providing personalization and navigation features that allow users to create the working environment that best fits the way they need to work.
- Facilitating communication and collaboration between people who need information and the people who can supply the information.

Knowledge portal products incorporate a combined collection of the features and functionality outlined by these individual portal market segments. Portal products collectively should be evaluated as possible applications for the organization to use to establish our enterprise portal or web portal solution. The features and benefits to look for in knowledge portal software solutions should include:

- A single point-of-access from a web browser to applications, business content, and services. We should, however, verify which web browsers and version numbers are supported by portal software applications. When multiple web browsers and versions are supported, they are not always supported with the same functionality or feature set.

- A personalized, role-based user interface that's customizable to individual needs.
- Simple maintenance that requires no additional client soft-ware or hardware.
- Access to enterprise applications and data sources using a development application programming interface (API).
- A single sign-on into the portal solution to access all the applications and data sources available in the portal desktop.
- Integrated communication and collaboration features with existing enterprise applications, systems, and hosted services.
- Bundled search engine or integration capabilities with third-party products to provide search features and functions.
- Bundled web-based business intelligence tools or integration capabilities with third-party business intelligence applications.

Portal software vendors use a combination of several programming interfaces and software functions or actions to implement the portal functionality in their products. Each of these portal software applications implements or supports portal features differently. The best process to select the appropriate portal software vendor is to determine the objectives and requirements of our enterprise or web portal solution and evaluate which product most appropriately matches our needs.

WEB PORTAL SOFTWARE FUNCTIONS

The web portal software functions are the core services, actions, and building blocks that allow the organization's self-service applications to be included in the web portal solution. The organization must review the software functions and determine how employees in the organization want to use these functions in their current processes and job responsibilities. The portal software application provides software functions that employees have not been able to take advantage of using their current business processes. In the process of reviewing the business functions and self-service applications for the web portal solution, we also want to consider how the software functions can be used to enhance or improve the organization's existing business functions.

Web portal software functions all contribute to creating a consistent view of the organization for every employee. By concentrating on how to implement these major software functions, the business processes and self-service applications used by employees are made available in the knowledge desktop unique to their role in these business functions. It is important that information that is presented be accurate and timely to provide the value necessary to make better decisions as an individual and more consistent decisions as an organization. The most common software functions that portal software solutions need to support include:

Data points and integration: This element provides one of the fundamental functions to be supported by the web portal solution, including the ability to access information from a wide range of internal and external information

sources and display the resulting information at the single point-of-access desktop. The organization needs to verify that our applications and data sources can be integrated into the portal software solution selected.

Taxonomy: The taxonomy or categorization element provides information context, including the organization-specific categories that reflect and support the organization's business. The taxonomy provides rapid recognition of common terminology used in the organization and improves semantics for users of the web portal solution. The taxonomy established for the organization should be logical and easily recognized or understood by employees that use the web portal. When evaluating portal software solutions, we should verify that the organization's naming and categorization standards can easily be implemented and maintained.

Search capabilities: The search element provides several services for the web portal users and needs to support searches across the enterprise, the World Wide Web, and third-party search engine catalogs and indexes. Users will want the portal solution to support search queries that can be run on a scheduled basis or as an immediate request. Make certain that the portal software solutions considered are able to support sophisticated search requirements and that the search element is flexible enough to meet future expectations as well as current requirements. As the portal solution grows in the organization, we can expect search requirements and functionality needed to expand.

Help features: The help element provides assistance when using the web portal solution. Help should be available for both web portal features and organization-

specific features. Several different types of help functions should be available from the web portal desktop. Our portal solution may require simple pop-up help when a mouse is held down over a menu option, context-sensitive help for information entered in a specific field, or detailed help on a specific topic or issue. The help function allows portal users to do some self-discovery on an issue before having to contact a knowledge expert to assist them or answer a question.

Content management: The publishing and distribution or content management element supports content creation, authorization, and inclusion in (or exclusion from) web portal content collections. As a rule, organizations have volumes of information and reference material to make available from the web portal. There are new policies, procedures, forms, templates, announcements, schedules, and much more that need to be pushed continually to employees. The content available from the web portal desktop should be available for distribution in multiple online, downloadable, or hard-copy formats. Content management and third-party integration features available in portal software products are important factors to review and match to the organization's requirements.

Process and action: The process and action element enables the web portal user to initiate and participate in the organization's business processes. Often there are two different implementation approaches to integrating workflow or process and action features in the web portal solution. The first is to integrate existing workflow solutions into the web portal desktop for users. The second is to design or script a solution using the development environment and tools available using the portal software product's programming interface. Make certain that the web

portal software selected by the organization can support how employees need to work to complete their assigned responsibilities.

Collaboration and communication: The collaboration and communication element facilitates discussion, locating innovative ideas, and recognizing resourceful solutions. This element gives employees the ability to work together in a qualitatively better way by creating a shared, virtual desktop (collaboration), supporting electronic messaging (communication), and adding collaboration and communication features to business processes (coordination). Portal software functionality needs to include easy configuration and administration features to existing enterprise messaging, personal information management, and calendaring systems.

Personalization: The personalization element is a critical component to create a working environment that is organized and configured specifically to each employee in the organization. Employees are empowered to work efficiently with the information needed to complete their jobs easily available. They can also work effectively by being allowed to define and organize their individual desktops. Decision-making capabilities are optimized to each portal user's working style and content preferences. The key is achieving balance between the features and information that need to be consistent and constant in the portal solution and the personalization and unique features needed by individual employees. Being able to control the content and structure of multiple desktops configured throughout the organization requires a portal solution that is easily configured and maintained. We need to research and review administration and configuration

features available in portal software solutions to verify that the IT department can develop, deploy, support, and maintain the personalization of our portal solution.

Presentation: The presentation element provides the knowledge desktop and visual experience to the web portal user that encapsulates all of the web portal's functionality. All of the software functions or elements mentioned previously must be supported in the presentation element in a format that is flexible enough to allow intuitive use and easy navigation throughout. The presentation element contains all the software functions or elements available from the web portal desktop. How this collection of software elements might be organized and displayed in a user interface is shown in Figure 2-2.

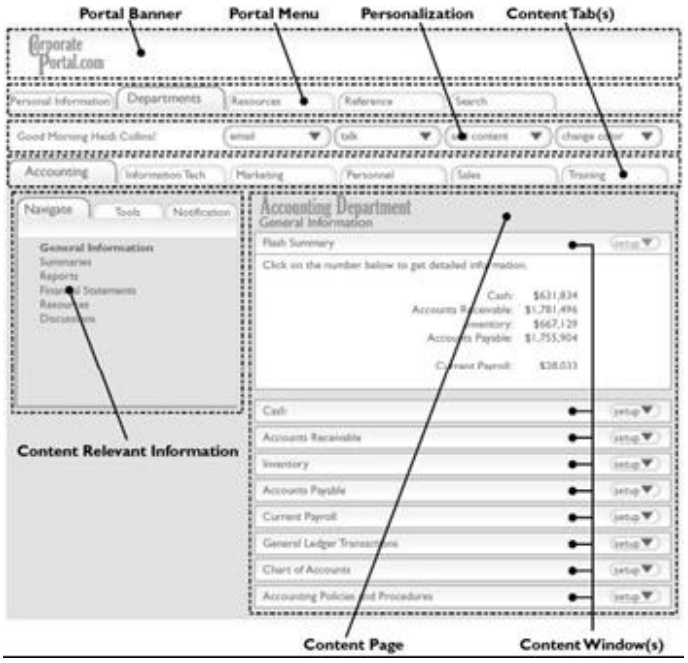

Figure 2-2: Main components of the web portal user interface.

Administration: The administration element provides two services. The first is the deployment and maintenance activities or tasks associated with the web

portal system. The second is what can be uniquely configured by an administrator for the web portal system as well as by each user through web portal personalization.

Security: The security element provides a description of the levels of access each user or groups of users are allowed for each portal application and software function included in the web portal. The types of access allowed include no access, reader access, editor access, or other. The organization's IT security requirements for intranet applications need to be addressed and supported by the portal solution implemented by the organization. Research and review the security models of third-party portal software applications to make certain the security model included in the portal software application meets the IT standards defined for the organization.

WEB PORTAL UNIVERSAL FEATURES

When all the software functions are integrated into a single desktop interface, there need to be additional portal features to monitor and control how these functions interact with each other. These universal features or control mechanisms are the consistent and pervasive user interface components that tie the software functions together in the web portal desktop. The organization must review this set of features to define how they can be implemented in the web portal application.

Initial collections of the universal features for the web portal need to be defined in the first phase of the project and documented in the business case. This list of

universal features can then be continually reviewed and enhanced as new phases of the web portal solution are implemented. These cohesive web portal universal features include:

Self-service applications (access to discrete enterprise data): All roles and responsibilities, processes and activities, and departments or workgroups that are elected to be part of the web portal solution need to be identified. The summary and scorecard information made available from the portal desktop usually originates in several different applications and systems throughout the organization. Additionally, we must establish the interaction or processes users will have available to them with the information presented in the portal desktop. Finally, we need to address usability of the information presented in the portal desktop. Users tend to want to be able to easily drill down into information, navigate to additional information or processes, leave the portal desktop to work in other applications when necessary, and return to the web portal desktop to continue their daily activities.

Web portal (aggregation of self service applications): Software functions and web portal universal features must behave consistently and be easily understood by users of the web portal self-service applications. To ensure this, the first step is defining or recognizing how these software functions will be incorporated or allowed to behave within each self-service application or staged delivery. The number of web portal users and the behavior of the software functions also need to be evaluated and possibly updated between staged deliveries to accommodate more users or expanded functionality. The organization should

create a main portal menu hierarchy that best outlines or defines how information can be located, which may be by a combination of responsibilities, activities, and departments. As an example, we might create an initial menu hierarchy system that works well for the first group of web portal users. All portal users can find information in less than four or five mouse clicks and have established a collection of bookmarks or favorite content pages that they can easily find. Over time, additional users and the information they need to access need to be included in the web portal solution. The organization must decide whether to reevaluate the portal menu hierarchy with additional levels, create a second or third menu hierarchy that users can select from, or establish a corporate menu hierarchy that everyone must use combined with a personal menu hierarchy that can be configured by each user as the solution. Alternatively, we may choose a unique solution created to meet the organization's specific user requirements. We need to verify that the portal software solutions we evaluate can easily support whatever navigation and configuration features we decide on for our web portal solution.

Comfort and familiarity (user personalization): Web portal users need to be able to customize and organize their knowledge desktop to work effectively from a single point-of access. Icons, logos, and colors can be useful to distinguish information associated with the organization. Individuals with similar responsibilities from personal information management (PIM) to specific business functions may require a consistent set of available features in the portal solution so they can work together effectively. However, this set of consistent features

needs to be supported with some level of individual personalization. This personalization might include allowing features or content to be rearranged on the desktop, bookmarks and favorites to be set or personal folders or menus to be maintained and updated. The comfort and familiarity goal is to allow the user to think and work as efficiently as possible while making better, more informed, and consistent decisions for the organization. The flexibility and extent of the user personalization requirements must be outlined by the organization, reevaluated on a scheduled basis, and enhanced as the web portal solution matures.

Understanding and knowledge (navigation personalization): Presentation of relevant information and navigation through that information is another global consideration to be evaluated and included in the web portal solution. To facilitate navigation through an extensive information network without overwhelming the web portal users in the organization requires the implementation of corporate knowledge maps. The process of creating knowledge maps that translate into a cross-reference of resources and information in the organization is a valuable exercise. This experience should not be looked at as a one-time event but a continual process of looking at the organization's knowledge from different perspectives to address current and future issues. The final result is a road map of the organization. If we have created a set of maps that describe all the available routes from one piece of information to all other pieces of information, then by knowing exactly where we are on the map, we are able to get to every other location on the map.

System intelligence (behavior assistance personalization): Change will be the most notable constant throughout the design and implementation of the web portal. Each employee has a definite and adjusting perception of what should be available in the web portal. Our portal software solution should help us implement two strategies with respect to behavior assistance personalization:

- Tracking web portal enhancements. As we are creating the web portal strategy, we may experience relentless or accelerating change. The web portal has to recognize and adjust to changing requirements in information needs throughout the organization. A successful web portal strategy should therefore incorporate a well-defined plan for evaluating and updating web portal software functions and universal features. Our web portal needs to include a learning process to adjust the parameters of each employee's requirements on a scheduled basis. This means that several of the set of consistent features and global elements included in our web portal solution must be reevaluated, enhanced, and expanded as the web portal solution is iteratively developed and deployed. The preliminary web portal strategy needs to outline the aspects of each self-service application, each software function, and the web portal universal features in order to document and provide change management procedures for each of these elements. These procedures allow the organization to continually recognize and adjust to ongoing change, in effect creating an intelligent system for the employees. Make certain that the portal

software solution implemented in the organization supports a comprehensive API for our internal development team to take advantage of.

- Finding a way to identify employee behaviors and activities and learn from them. We can create feedback processes and opportunities within the portal desktop and begin to use the results to help employees. Amazon.com uses this type of functionality to direct customers toward additional services and products that match previous buying preferences. The organization can use this same type of pattern matching and cross-referencing of information and knowledge experts to provide additional web portal benefits for the employees.

WEB PORTAL USER INTERFACE

There are several layouts and implementations of the portal interface that combine all the components of the web portal software functions and universal features. One example of these user interface components is shown in Figure 2-2. The components available include:

Portal banner: A graphic provides a visual representation of the page represented in the web portal.

Portal menu: A persistent menu provides access to applications, functionality, and services throughout the organization.

Personalization: A persistent set of options is presented that allows users to modify the look and feel of the web portal user interface.

Content page tab(s): A collection of secondary menu options that are associated or related to the context of the current content page are also presented. These menu options provide access to additional content pages.

Content page: This is the information presented in the main window of the browser associated with the portal menu option or other navigation option selected by the web portal user.

Content window(s): The collection of information contained in the content page can be a single window or several windows of information contained in the same content page.

Content-relevant information: Several additional features may appear in the content relevant information section of the web portal for easy access by users. These include:

- Tools (e.g., to access collaboration, communication, and applications that are standard in the organization)
- Navigation (e.g., to provide additional navigation options, including both data and knowledge experts, that are related or relevant to the information presented on the content page)
- Notifications (e.g., to provide important messages that are related to the context of the content page and need to be brought to the attention of the web portal user)

Portal software functions are integrated into the portal desktop in several different ways. All the portal software functions available in the desktop portal comprise

the overall presentation of the portal desktop. The portal banner provides presentation features for branding and visual recognition by employees. The portal menu allows the organization to create a standard taxonomy with semantics that employees can easily identify with. Users have direct hyper-links to topic help and search functionality through the portal menu. Employees have individual recognition and persistent access to all PIM applications, portal configuration features, bookmarks and favorites, and other employees through instant messaging from the personalization section of the portal desktop, which is also a good location from which users can easily participate in discussion forums throughout the organization. Each content page available in the portal desktop provides integration to information and applications throughout the organization. The portal software functions included in the main content page are data points and integration, content management features, process and action functionality, search functionality, and detailed help features that are available. There will also be personalization features available for users to interact with portal content pages. The content-relevant information is a dynamic section of the portal desktop that provides behavior assistance, additional navigation, and additional awareness of specific information that is of importance to the context of what is presented in the content page.

Chapter 3: Web Portal Strategy

IMPORTANCE OF WEB PORTAL STRATEGY

Creating and planning a web portal strategy is not easy. The job is to collect and analyze the information needed to formally propose a web portal solution. The process can be approached in a systematic way to increase the probability of success. We need to determine if the organization has several identifiable and measurable business reasons to build a web portal solution. Consider whether these problems exist in the organization:

- Employees in the organization need consistent information to make routine decisions.
- Employees need information from many different systems to make strategic decisions.
- Employees must complete most company-related activities online.
- Employees need to be able to access company information from an intranet site through a browser.
- Employees must access company information from multiple data sources and applications to complete their assigned activities.
- Employees cannot locate or navigate efficiently through company information or the intranet site.
- Employees are not aware of information or resources available in the company to complete their responsibilities.
- Employees require extensive training to use portions of the applications and systems they need to use to complete their responsibilities.

If our first impression is that a web portal solution would provide several benefits or resolve several known issues for the organization, then the development of a web portal strategy and the completion of additional analysis activities would be beneficial.

THE WEB PORTAL STRATEGY

The web portal strategy is a collection of identified activities that, once completed, determine the feasibility of implementing a web portal solution in the organization. In most cases a good place to start is (1) to evaluate our information technology (IT) environment and (2) identify the most critical business challenges in the organization. We should consider first the business functions and self-service applications selected for implementation in the web portal solution from both the IT and end-user perspectives. Once the initial ideas have been selected, the business processes are outlined and presented as a series of screens that represent a visual model or storyboard of the proposed web portal solution. Other activities that need to be completed as part of the web portal strategy are an analysis of the financial impact and a project development timeline to deploy the web portal. This collection of activities is described as the discovery process to help identify, prioritize, and define the business functions, self-service applications, and web portal features to document in the business case. The activities within this process include:

- Information technology review
- Preliminary identification of web portal business functions

- Storyboarding
- Financial analysis
- Creation of a web portal project plan

The objectives of our web portal strategy include the following:

- Defining a high-level overview of web portal concepts and technologies.
- Understanding the IT environment.
- Identifying and defining the highest-level business requirements.
- Determining the web portal solution overview.
- Defining the web portal solution storyboard.
- Defining the financial analysis.
- Defining the preliminary project plan.
- Building the web portal business case.

The documentation deliverables of our web portal strategy include:

- Business case document (i.e., the web portal proposal)
- IT assessment
- Preliminary storyboard
- Financial metrics analysis
- Preliminary project plan

WEB PORTAL CONCEPTS AND TECHNOLOGIES

We want to outline the primary strengths of the web portal solution that provide the inspiration for a successful implementation. What are these strengths? They may be unique for the organization. Combining knowledge of web portal software functions and the web portal universal features with the knowledge we have regarding the initiatives and requirements of the organization should provide enough information for us to evaluate and identify the web portal strengths and benefits. Web portals are designed to improve decision cycles in the organization and can incorporate knowledge sharing into all decision making activities completed by employees. The web portal solution we implement should improve how employees gather information, research and collaborate, and take action.

A web portal solution must be approached and built in a hierarchical fashion. There are at least three generations of requirements that continue to build on each other when implementing a web portal solution. The three generations of web portals and the universal features and objectives of each are as follows:

Implementation of Web portal	**Objectives/Features**
First generation	Build and present the core content Identify and develop self-service applications (i.e., access to discrete enterprise data)

Second generation	Extend the solution to all intranet activities Develop the web portal (i.e., aggregation of self service applications)
Third generation	Create a personalized environment with focus on: Comfort and familiarity (i.e., user personalization) Understanding and knowledge (i.e., navigation personalization) System intelligence (i.e., behavior assistance personalization)

An agreed-upon web portal definition should be comprehensive enough for anyone reading the business case to understand the purpose of the solution being proposed. The definition should also be concise enough to provide direction for the organization. Over time, the organization may change its web portal definition, which may lead to new opportunities. As employees in the organization begin to use the web portal and the web portal environment becomes more complex and competitive, we may want to expand the definition. For the purpose of the first-generation web portal solution and its business case, the following questions must be answered by our web portal definition:

THE IT ENVIRONMENT

A review of existing initiatives in the IT department or other departments needs to be completed. This is accomplished by reviewing projects in development or those scheduled to be developed throughout the organization. After completing this analysis, we may want to consider integration of the web portal solution as a component of an existing initiative. This might prove to be more cost-and budget-effective than having to establish a new project initiative or locate new funding and budget sources for the web portal solution. The web portal could prove to be easily included in intranet, extranet, or knowledge management initiatives.

The more information we uncover or become familiar with about the organization and the IT department, the more effective we will be in managing and planning a web portal solution business case. A review of the IT infrastructure must be conducted as part of the discovery process. The purpose of this review is to determine how to take advantage and leverage the existing IT infrastructure as part of the web portal solution. It is also critical to understand where technical limitations exist and what improvements of existing enterprise systems or infrastructure need to be included as part of the web portal business case. This documentation is collected from the IT department and business domain experts to provide insight into the strategic direction, business processes, goals, and critical success factors for the web portal solution being proposed in the business case.

There are many IT projects and administrative duties that contribute to a web portal solution. The intranet, data warehouse, relational database management system, reporting system, enterprise resource planning (ERP) system, financial system, messaging system, and multitude of other systems and applications—all provide data and user interface screens to employees from the web portal desktop. As self-service applications are added to the web portal, we must verify that the data sources and appropriate applications are fully implemented. The IT department must already have complete implementation and support procedures available for the information or application that needs to be included in the portal desktop. The portal desktop only serves as a window or gateway to existing data sources, systems, and applications.

BUILDING THE WEB PORTAL BUSINESS CASE

Only when all the web portal strategy activities have been completed and our final analysis is done we will have the information needed to explain the real demand for a web portal by employees throughout the organization. The final analysis and the feasibility study are documented as a web portal business case. This document includes an explanation of the demand and an associated cost structure that yields a sufficient return on investment (ROI) to justify the investment in time, human resources, and money that must be made by the organization to deliver a web portal solution. The time investment is presented as a project timeline and plan. The web portal project plan must address the planning, architecture and design, development, and deployment activities for the

first phase of the web portal solution. The business case is based on preliminary decisions coming out of the research and analysis activities. The actual web portal project will require the consensus and approval of the user community, business domain experts, and project sponsors regarding the business functions, self-service applications, IT environment, and software applications before development of the web portal solution begins.

A complete business case should be prepared that provides a description of the web portal solution and its impact on the organization. The research required to build the business case provides most of the information required to prepare the documentation. The business case must be specific and well written. Once the business case has been completed, the web portal strategy team must consider several presentation methods. They range from printed copies to be distributed to PowerPoint presentations, a demonstration, collaborative discussions, and question-and-answer sessions, among other methods. First, it should be recognized that all plans are subject to change depending on circumstances and the passage of time. What was a reasonable case last quarter may no longer be reasonable this month. The business case documentation should therefore be version controlled so that it can be updated on an as-requested basis.

The purpose for creating a business case is to build a proposal for a web portal that can convince the organization to move forward with this initiative as a knowledge management solution. The business case documentation should be used as a tool to support the planning and decision-making activities required to move forward with a web portal. The business case documentation outlines the

infrastructure requirements, the self-service applications and features requirements, suggested standards and best practices, discounted cash flow, payback period, and internal rate of return that a web portal solution can offer the organization. As we collect information and complete the research and analysis activities, we can reach conclusions regarding the financial and other business consequences of a web portal solution. Once we have determined that a web portal business case is a necessary tool to present the web portal as a knowledge management initiative to decision makers in the organization, a wide spectrum of detailed information needs to be collected from almost every source in the organization. The process may seem overwhelming. There are five fundamental steps to be completed:

- Laying out the basic web portal concepts.
- Gathering data on the feasibility and specifics of the web portal concepts.
- Focusing and refining the web portal concepts based on the data compiled.
- Outlining the specifics of our proposed web portal solution.
- Putting our business case in a compelling format.

INFORMATION TECHNOLOGY ANALYSIS AND RESULTS

The ability to implement a web portal solution or any other knowledge management initiative depends on the information technology (IT) department's contribution of knowledge, time, and resources to the project. A survey should be

created by the web portal strategy team, and submitted for the IT department to complete. Once we have worked with members of the IT department to complete the IT interview questions, we must analyze the results for our internal planning purposes, coordinate that information, and document the results. The summary we create is then added to the web portal business case. The following checklist is useful:

- Background
- Messaging infrastructure
- Collaboration systems
- Intranet strategy
- Internet strategy
- Search systems
- Online help systems
- Content management systems
- Enterprise resource planning (ERP) systems
- Line of business (LOB) systems
- Legacy applications
- Data warehouse and data mart systems
- Business intelligence and executive information systems
- Knowledge management systems
- Cultural considerations (acceptance of the web portal solution by employees)
- Web portal management

- Administration
- Technical staff and support

Chapter 4: Portal Design and Development Guidelines

Introduction

This chapter focuses on the design approaches for portal solutions. We try to cover some of the differences between traditional Web applications and portal solutions from a design standpoint. We provide guidelines on how to design portlet applications in WebSphere Portal V5 using different types of frameworks, and discuss best practices for portlet development. We also include information concerning performance, both in regard to portlet design and development, and also in regard to response time tracing within a portal solution.

In addition, we shall see how to add collaborative functionality to our portlet applications and introduction to an approach to integrate and reuse existing business applications within the Portal.

WebSphere Portal

WebSphere Portal is quickly becoming a standard infrastructure platform for building enterprise-class applications. But it provides much more than just this:

WebSphere Portal delivers integrated content and applications, plus a unified, collaborative workplace.

Thus we can look at WebSphere Portal from two different angles. On one hand it is a development platform for new applications, and on the other it enables the integration of existing applications and processes. We reflect on both aspects and discuss patterns and strategies that will help to build successful Portlet applications.

Introduction to object-oriented design patterns

We begin by looking at object-oriented design patterns that are useful in the Portlet development process.

Design patterns are common strategies for developing reusable object-oriented components. Each pattern describes a problem which occurs over and over again in a particular environment, and then describes the core of the solution to that problem in such a way that this solution can be used a million times over, without ever doing it the same way twice. We now consider patterns for components design. The patterns that we show here can be used in other solutions and in other programming languages.

These patterns are divided into three categories: creational, structural, and behavioral.

Creational

These patterns are those that create objects, rather than having to instantiate objects directly. This gives a program more flexibility in deciding which objects need to be created for a given case.

Structural

These patterns help to compose groups of objects into larger structures, such as complex user interfaces or accounting data.

Behavioral

These patterns help to define the communication between objects in the system and how the flow is controlled in a complex program. The patterns described are:

Singleton, Factory, Abstract Factory, Proxy, Decorator, Command, Facade and Model-View-Controller (MVC)

Singleton

This is a creational pattern that is used to ensure that a class has only one instance, and to provide a global point of access to it. This pattern is interesting when we want to keep track of a sole instance. We can use this in many ways, for example, when we want to load application variables from a file or control the access to components. The easiest way to make a class that can have only one instance is to embed a static variable inside the class that we set on the first instance and check each time we enter the constructor. A static variable is one for which there is only one instance, no matter how many instances there are of the class.

static boolean instance_flag =false;

The problem is how to find out whether or not creating an instance was successful, since constructors do not return values. One way would be to call a method that checks for the success of creation, and which simply returns some value derived from the static variable. Another approach, suggested by Design Patterns, is to create Singletons using a static method to issue and keep track of instances. To prevent instantiating the class more than once, we make the constructor private so an instance can only be created from within the static method of the class.

Factory

This is a creational pattern that is used to define an interface for creating an object, but lets subclasses decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses. This approach can be found in EJB technology for home and remote classes. A Factory pattern is one that returns an instance of one of several possible classes depending on the data provided to it. Usually, all of the classes it returns have a common parent class and common methods, but each of them performs a task differently and is optimized for different kinds of data.

Abstract Factory

This is a creational pattern that provides an interface for creating families of related or dependent objects without specifying their concrete classes. This approach can be found in EJB technology for home and remote classes. The Abstract Factory pattern is one level of abstraction higher than the Factory pattern. We can use this pattern when we want to return one of several related classes of objects, each of which can return several different objects upon request. In other words, the Abstract Factory is a factory object that returns one of several Factories.

Proxy

This is a structural pattern that provides a surrogate or placeholder for another object to control access to it. The Proxy pattern is used when we need to represent a complex object with a simpler one. If creating an object is expensive

in terms of time or computer resources, Proxy allows us to postpone this creation until we need the actual object. A Proxy usually has the same methods as the object it represents, and once the object is loaded, it passes on the method calls from the Proxy to the actual object. This approach can be found in remote implementation of EJB technology.

Decorator

This is a structural pattern that attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub classing for extending functionality. The Decorator pattern provides a way to modify the behavior of individual objects without having to create a new derived class. Suppose we have a program that uses eight objects, but three of them need an additional feature. We could create a derived class for each of these objects, and in many cases this would be a perfectly acceptable solution. However, if each of these three objects requires different modifications, this would mean creating three derived classes. Further, if one of the classes has features of both of the other classes, we begin to create a complexity that is both confusing and unnecessary. We can see this applicability in the portlet API. We have one skin for each portlet and the skin has some functionality such as resizing the portlet, call edit mode for personalization issues, and so on.

Command

This is a behavioral pattern that encapsulates a request as an object, thereby letting us parameterize clients with different queue or log requests. The

Command pattern forwards a request to a specific module. It encloses a request for a specific action inside an object and gives it a known public interface. It lets us give the client ability to make requests without knowing anything about the actual action that will be performed, and allows us to change that action without affecting the client program in any way.

Facade

This is a structural pattern that provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use. Usually, as programs are developed, they grow in complexity. In fact, for all the excitement about using Design patterns, these patterns sometimes generate so many classes that it is difficult to understand the program's flow. Furthermore, there may be a number of complicated subsystems, each of which has its own complex interface. The Facade pattern allows us to reduce this complexity by providing a simplified interface to these subsystems. This simplification may in some cases reduce the flexibility of the underlying classes.

Model-View-Controller (MVC)

A design pattern particularly useful for Portlet applications is the Model-View-Controller or MVC pattern. A number of different types of skills and tools are required to implement various parts of a Web application. For example, the skills and tools required to design an HTML page are vastly different from those required to design and develop the business logic part of the application. In order

to effectively leverage these scarce resources and to promote reuse, we recommend structuring Web applications to follow the Model-View-Controller design pattern.

In the design shown in Figure 1, Model represents the application object that implements the application data and business logic. The View is responsible for formatting the application results and dynamic page construction. The Controller is responsible for receiving the client request, invoking the appropriate business logic, and based on the results, selecting the appropriate view to be presented to the user.

The Model represents enterprise data and the business rules that govern access to and updates to this data. Often the Model serves as software approximation to a real-world process, so simple real-world modeling techniques apply when defining the Model.

A View renders the contents of a Model. It accesses enterprise data through the Model and specifies how that data should be presented. It is the View's responsibility to maintain consistency in its presentation when the Model changes. This can be achieved by using a push Model, where the View registers itself with the Model for change notifications, or a pull Model, where the View is responsible for calling the Model when it needs to retrieve the most current data.

A Controller translates interactions with the View into actions to be performed by the Model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a Web application, they appear as GET and POST HTTP requests. The actions performed by the Model include activating business

processes or changing the state of the Model. Based on the user interactions and the outcome of the Model actions, the Controller responds by selecting an appropriate View.

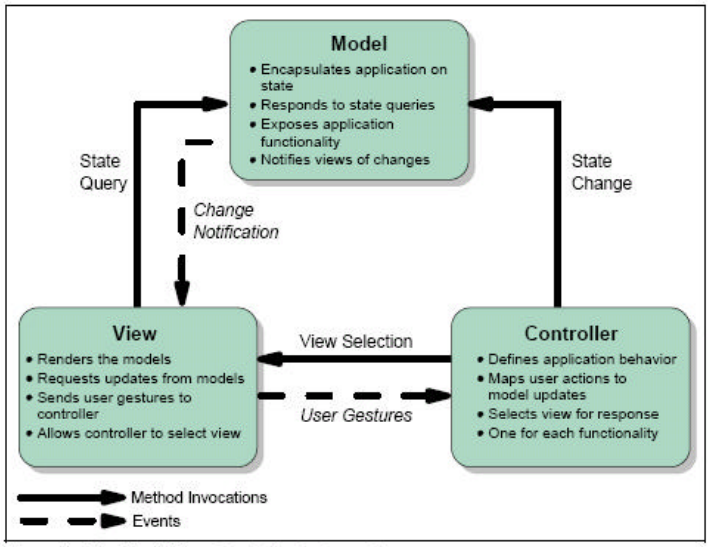

Figure 1 The Mode-View-Controller design pattern

Introduction to the Portlet framework

A Portal provides a user with a consistent view of portal applications (Portlets).

The portal allows the administrator to define specific sets of applications, which are presented to the user in a single page context. The Portlets themselves are more than simple views of existing Web content. A Portlet is a complete application having multiple states and view modes, plus event and messaging capabilities. Portlets run inside the Portlet container of a portal server, similar to a servlet running on an application server. The Portlet container provides a runtime environment in which Portlets are instantiated, used, and finally destroyed.

Portlets rely on 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. The user interacts with Portlets through Action-URIs. By clicking on a link inside of the Portlet the user can perform certain actions, for example, changing the display view. Internally the Portlet stores persistent information about its current state. One Portal page aggregates many different Portlets at the same time, which is an important issue to consider when we design and work with Portals.

Navigational schema

At first glance, the navigation inside of a Portal is comparable to the navigation of a simple Web page. This is true as long as the user navigates between Portal pages. However, the situation gets more complex as soon as the user starts interactions with a Portlet. From the browser's point of view, interactions with a Portlet result in loading of a new page. This is correct in that the URL changes.

From a user's point of view, he still seems to be on the same page. Only the small area of the Portlet itself changes.

This usage pattern of the Portal becomes interesting as soon as the user presses, for example, the back button. The browser will revert back to the previous URL, but nothing will change on the screen. This happens because the previous URL does not contain an action that tells the Portlet to revert back to its previous state. The user probably would have expected that the last Portlet action would be undone.

We see that navigation in the Portal has two different perspectives: first, the aspect of navigation between pages, and second, the application-internal navigation. The browser's back and forward buttons can only be used to navigate between Portal pages. Portlet internal navigation needs to be implemented as Action-URIs. Otherwise the persistent states inside of the Portlet can't be managed.

General approach to portlet development

Portlet developers typically have relied on the portlet API as the basis for their implementation. This API exposes the components needed to develop fully functional Portlets. We can consider the portlet API itself as a framework that we can use to develop Portlets. In fact, it is not really an option not to use it; we must use if we want to develop Portlets for WebSphere Portal V5. The portlet invocation API is very similar to the Java servlet API. The portlet class corresponds to the servlet class, with method signatures matching the init, service, and other key methods. Similarly, the PortletConfig class corresponds to ServletConfig, PortletContext corresponds to ServletContext, PortletRequest corresponds to ServletRequest, and so on.

It provides us with a super class PortletAdapter from which we extend to develop our own portlet. We can choose to implement the doView(), doEdit(), and the other methods, inherited from the PortletAdapter class, to provide a portlet with some functionality. In an MVC pattern this portlet class becomes our controller.

In our controller we can then retrieve the data from all the different data sources we have identified. This can be done using a number of integrator technologies.

We can then have our collected or business logic data displayed by use of JSPs. This is done by releasing the control from the portlet to the JSP through the PortletContext; we are now able to transfer our data by using, for example, viewbeans and the PortletRequest object. However, the portlet API does not provide assistance to implement a well-designed page navigation model. When creating more complex Portlets, a developer must handle the view navigation within the portlet, and consider how to efficiently use action events and listeners to respond to user action events, such as clicking on a button or a link.

State pattern portlet framework

Without a well-designed approach to this problem of how to best implement control logic, we end up creating Portlets that contain a fair amount of code dedicated to simply addressing the intent of a user's request. Besides being repetitive, this code makes the portlet much harder to read since we need to wade through the control logic to get to the real work that the portlet is doing. The control logic is also prone to error because it relies on various techniques, such as string name matching, to hook an event to a listener or a listener action to behavior in the portlet method. It can also complicate the business logic since the implementation of the control function can resemble the business logic. To solve these problems with control logic, we can consider our application to be a collection of portlet actions and states. Then, given a well-defined approach to state transitions, we can remove the cumbersome control logic code from within the portlet application.

Problems with the standard implementation approach

A simple MVC-based portlet may present a list of items that are retrieved from a database and then presented to the user. When the user selects an item, a detailed view is displayed showing specific information about the item selected.

This might involve another request to the database. The important point here is that a requirement now exists to follow two different control flows. In the first step, the controller must call the business objects to create the appropriate beans, which are passed to a JSP to be rendered in the main view. In the second step, the controller retrieves an indicator of the selected item from the user's request and then calls the business objects to create the appropriate bean. Which will then be passed to a different JSP to render in a detailed view, separate servlet implementation, and then we have to use a framework, such as Struts, to implement a similar MVC design.

The components of a state pattern implementation

Applying the State pattern to Portlets lets developers cleanly implement process control. The State pattern uses the following components:

StateManagerPortlet

This is the main portlet class. It is portlet-independent, and is where we would typically write all of the portlet-specific code. The class serves as a dispatcher to support action and state classes where the portlet code resides. StateManagerPortlet implements the actionPerformed method and the doView, doEdit, doHelp, and doConfigure methods. The actionPerformed method simply gets the current instance of the action class and dispatches to its

actionPerformed method. Similarly, the do methods get the current state object and ispatch to its performing method. Therefore, the StateManagerPortlet does not need to know anything about the current portlet implementation and does not have a large number of ifs and checks to determine where processing should continue. As long as we are familiar with the flow between states and actions, the processing will occur properly, without us having to code much extraneous control logic.

Action

Classes that implement this interface will implement an actionPerformed method. This method performs any action necessary to implement the function required by the action request. However, the function implemented is specific to the action event being invoked. An individual actionPerformed method for a specific action class contains code only for that particular action. This method also sets the current state for further processing. In this flow process, an action gets called, it performs work specific to its function, and then sets the state for the next transition.

State

Classes that implement this interface will implement a perform method. This method gets called from the do methods of the StateManagerPortlet, and contains the code that would normally reside in these methods. Again, this code is specific to the state of the class it resides in, and therefore avoids additional clutter. Typically, the state's perform method will invoke a JSP to render its results. The UI might let the user set other actions within the portlet. The JSP

associates an action class with each of these actions on the page. When a user invokes one of these actions, the appropriate action class instance is invoked from the StateManagerPortlet actionPerformed method, and state transition occurs. The state class is not responsible for state management and transition.

InitialStateManager: This class provides the initial states for each of the supported portlet modes.

ActionClassManager: This class provides the mapping from the given string to an Action class instance.

Actions and states

Applying the State pattern results is a cleaner implementation. Also, as we move between portlet modes, the state is always remembered. When we use the actionPerformed method to retrieve form data and hold it in session, we should avoid any problems with form data not getting reposted when the portal page gets refreshed. With this pattern, we can easily determine where and when we want data to be retrieved from the source, and when it should be retrieved from cache. Since actionPerformed methods do not get invoked on portal page refreshes, we can place data access codes in the action state and cache them there. The state classes may use data from cache to avoid multiple trips to get data when the portal page is refreshed.

Implementation specifics

Consider the following scenario. We have a portlet that presents a list of items on the main page to the user; the user can select one item to get more detailed

information (which is presented on another page). The user is also able to add, edit, and delete items. The data is persisted in a database. In this example, the items are contacts, as in an address book. Displays a similar form with the existing data inserted for modification In this case, we do not have an explicit page to delete an entry. The behavior is such that the user can select a contact entry to delete from the main edit page. There will be no confirmation page or successful execution page. Entry deletion processing occurs and the main edit page is refreshed. If an error occurs, an appropriate message will be displayed, but under normal processing, there isn't a view associated with this action. Assume that we want to have the Main and Detail view pages available to users with view permission for the portlet, and add, edit, and delete capabilities available to users with edit permission for the portlet. In this case, the Main and Detail view pages will be controlled from the doView portlet method. The remaining pages will be controlled from the doEdit method.

Using the standard approach

First, consider the doView method. Using standard portlet programming techniques, we can implement as in Example 1.

Example 1 doView

```
public void doView(PortletRequest request, PortletResponse response)
  throws PortletException, IOException {
  String oid = request.getParameter("selectedContact");
  If (oid == null) {
      // Main view processing goes here
      portletContext.include("main_view.jsp", request, response);
  else {
     // Detail view processing goes here
     portletContext.include("detail_view.jsp", request, response);
   }
 }
```

However, we realize that if we attempt to retrieve the selectedContact index from the HTTP request on every refresh, then when our portlet gets refreshed as a result of the portal page getting refreshed, the form data is lost and this implementation returns the portlet to the main view—even if the user was working with a different portlet on the page. Therefore, we use an action listener to get the selected contact value and set it on session so that future refreshes can get the correct value and preserve the appropriate page setting. Now the doView code looks like the code in Example 2, and there is an implementation for the actionPerformed method, Example 3, in the action listener that retrieves the appropriate value and puts it on session. We also need to specify the action event on the portlet URI so that the listener gets called.

Example 2 doView

```
public void doView(PortletRequest request, PortletRespose respons e)
  throws PortletException, IOException {
  PortletSession session = request.getPortletSession(false);
  if (session != null) {

    String oid = (String)session.getAttribute("oid");

    if  (oid == null) {
       // Main view processing goes here

      // URI and action listener for showing the contact details
      PortletURI portletURI = response.createURI();
      portletURI.addAction("detailRequest");
      request.setAttribute("detailURI", portletURI.toString());

      portletContext.include("main_view.jsp", request, response);

    else {
       // Detail view processing goes here
      // Get bean for given OID

     session.removeAttribute("oid");
     portletContext.include("detail_view.jsp", request, res ponse);
   }
 else
     response.getWriter().println("You must log in first");
}
```

Example 3 actionPerformed

```
public void actionPerformed(ActionEvent event) throws PortletException {
        String actionName = event.getActionString();
        PortletRequest request = event.getRequest();

         // Show the contact detail view. Get the contact oid number
        // of the selected contact and put it on the session object
        if (actionName.equals("detailRequest")) {
            String oid = request.getParameter("selectedContact");
            Request.getSession(true).setAttribute("oid", oid);
        }
}
```

There is quite a bit of code here to manage simply the page transitions between the main view and detail views, and we haven't even written any code yet for the application's business logic. There are many opportunities for bugs to be introduced into this code. Also, we need to implement component pieces in various places to make the process flow successfully; often these components rely on string matching to hook up an event with an appropriate action or set and read a flag in the code that manages process control. Including the additional actions to add, edit, and modify the control code would make the code more cumbersome. The code in Example 4 shows the actionPerformed method for this implementation. This does not display the corresponding code in the doEdit method that needs to set portletURI with the appropriate action, interrogate the event flags, and set and remove data and flags from session.

Example 4 actionPerformed

```
public void actionPerformed(ActionEvent event) throws PortletException {
        String actionName = event.getActionString();
        PortletRequest request = event.getRequest();
        PortletSession session = request.getPortletSession();

        // Show the contact detail view. Get the contact oid number
        // of the selected contact and put it on the session object
        if  (actionName.equals("detailRequest")) {
            String oid = request.getParameter("selectedContact");
            session.setAttribute("oid", oid);
        }
```

```
        // Handle the request to go to the add view
        if (actionName.equals("ADD_REQUEST")) {
            session.setAttribute("NEXT_EDIT_PAGE", "ADD_PAGE");
        }

        // Handle the request to go to the edit view
        if (actionName.equals("EDIT_REQUEST")) {
            session.setAttribute("NEXT_EIT_PAGE", "MODIFY_PAGE");
        }

        // Handle the  request to delete content
        if (actionName.equals("DELETE_REQUEST")) {
            ContactsManager.getInstance().(deleteContact(request));
        }

        // Handle the Add content event
        if (actionName.equals("ADD_CONTACT")) {
            ContactsManager.getInstance().addContact(request));
        }

        // Handle the Modify content event
        if (actionName.equals("EDIT_CONTACT")) {
            ContactsManager.getInstance().modifyContact(request));
        }
}
```

How does this change with the state pattern?

How can we improve this implementation? First, keep in mind that this application represents a collection of application actions and states. An action is a class that implements the action interface, and literally handles the processing for a specific application task or action. This is a portion of application code that exists in the actionPerformed method of the actionListener class; only this portion is specific to a single action event.

A state is a class that implements the state interface and represents the effect of the portlet as a result of applying an action. This class typically has a visual component.

Such an application would typically contain the following:

Actions:

- Show Main Page
- Show Detail Page
- Show Main Edit Page
- Show Add Contact Page
- Add a Contact
- Show Modify Contact Page
- Modify a Contact
- Delete a Contact

States:

- Main View Page
- Detail View Page
- Main Edit Page
- Add Contact Page
- Modify Contact Page

Now, determine the state transitions available for this application. The transitions are managed by applying the appropriate actions to the current state, which results in the application entering another specific state.

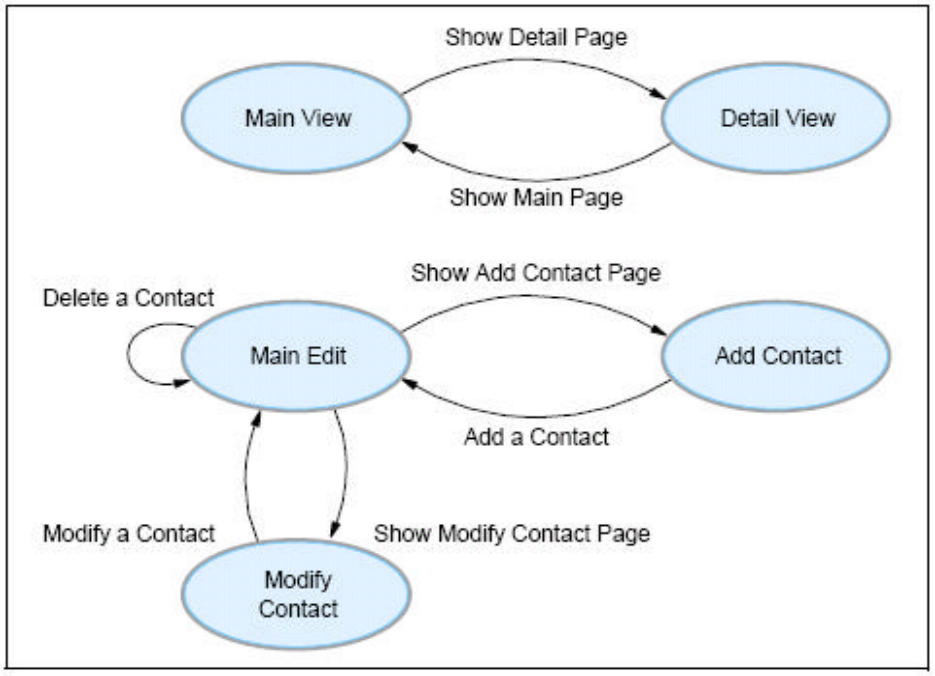

Figure 2 A state diagram

With this understanding of the portlet, we can create the application using the State pattern. The application is built using a common structure for state transition that lets us eliminate the excessive if statements with string matching and multiple flags are being set.

How does the state pattern work?

The code in Example 5 shows the simple processing in the doView method in the StateManagerPortlet class. The other do methods would be similar. We could easily collapse this processing to the service method. In this case, we would look for the state object from session using the portlet mode as the key. It is displayed this way since we would typically code to the do methods and the comparison would be more familiar. Also, we would extract the default state into a properties file or another initialization parameter to remove the connection to this class.

Example 5 StateManagerPortlet doView() method

```
public void doView(PortletRequest request, PortletResponse response)
        throws PortletException, IOException {
        // Ensure we are logged in
        PortletSession session = request.getPortletSession();
        User user = request.getUser();
         if (session == null || user == null)
            throw new MessageException("Login to the portal first");
            // Get the portlet state object from session, if not there get the initial
            // state for the mode.
            State nextState = (State)session.getAttribute(request.getMode().toString());
            if (nextState == null)
              nextState = InitialStateManager.getInitialState(request.getMode());

             // Dispatch to state handler
             nextState.performView(request, response, getPortletConfig().getContext());
 }
```

The code in Example 6 shows the simple processing in the actionPerformed method of the StateManagerPortlet class. Again, this method simply gets the action class instance identified by the action name string and invokes its actionPerformed method.

Example 6 StateManagerPortlet actionPerformed() method

```
public void actionPerformed(ActionEvent event) throws PortletException {

    // Execute the perform action method for the event
    PortletContext portletContext = getPortletConfig().getContext();
    PortletRequest request = event.getRequest();
    // Get the action handler class and dispatch
    String actionClassName = event.getActionString();
    Action action = ActionClassManager.getAction(actionClassName);

    // Dispatch to that class event handler; register the next state
    action.actionPerformed(request, portletContext);
    action.setState(request);
}
```

Then, the actionPerformed method of the action classes performs normal processing and also puts the proper state on the session for appropriate flow control. Keep in mind that the action and state dasses, in general, have either

very few fields or none. So, there is minimal expense in adding them to the session object.

Since no changes are needed to implement this pattern in the portlet do methods, we do not show the state classes perform method equivalents. However, we may extend the pattern here to include a refresh method for the state. The refresh method would logically extract the application code that is responsible for portlet data retrieval, so that we can isolate the logic needed to determine if a portlet refresh causes the source data to refresh or be retrieved from a cache.

Before the state class is asked by the StateManagerPortlet to write a markup fragment to the response (or invoke a JSP to do this), the StateManagerPortlet can ask the state class to refresh itself. In the refresh method, we can include logic to refresh or re-retrieve data for the display. For example, a portlet that displays volatile data may need to go back to the data source each time the screen is refreshed. We would include the logic to do this in the refresh method. On the other hand, a portlet that displays stable information would more likely put the data access logic in the initial state lazy initialization logic and cache the results. The refresh method for this type of stable data state would do nothing. Since business logic in the refresh method could cause a transition to a different state (including an error state), we could extend the state pattern to allow for state transitions from within states.

Struts

Developers may also choose to implement their portlets using the Struts framework. Struts is a Jakarta project providing a very popular open source framework for building Web applications. Using the Struts portlet framework provided with WebSphere Portal, Portlets written using the Struts framework can be run in WebSphere Portal. Struts provides considerable Web application functionality beyond page navigation and is an excellent choice for portlet development. The Struts framework is intended to provide a portlet application developer with two main advantages, specifically:

A controlled logic to our portlet applications.

The development of portlet applications requires the developer to coordinate several technologies to work together. In our examples we have seen that all requests are sent to the portlet class doView() method, and depending on the request and state of the Portlets, it redirects to the appropriate JSP pages for rendering. This is accomplished with a redirection map embedded in the portlet class. Basically, it's a tightly coupled application, since any change in the control logic would require recompiling and making sure that our application doesn't loose its integrity. The Struts framework solves this problem, especially in big and complex portlet applications, since it takes out the control logic map from the portlet class and places it on a flexible deployment descriptor, giving us a loosely coupled structure. Also, this map can be edited with graphical tools like the ones included in WebSphere Studio, the Struts tools. So the developer should consider using the Struts Portal framework if he is placing control logic into the

portlet applications and requires flexible and manageable control logic for the portlet.

A strong foundation for portlet development.

The Struts framework incorporates proven development patterns, like the MVC model 2 that will make our portlet application perform and scale appropriately. So if we are considering introducing a hybrid approach in the development of our Portlets, and we want to provide a strong structure, Struts can give us exactly what we need. The Struts framework will add some development components that have to be well understood and properly used in order to exploit its full advantages. So if our portlet application will just expose the control logic of our applications, it would be appropriate to reconsider using this framework. Also bear in mind that the standard Struts framework is accommodated to work in the WebSphere Portal infrastructure through the Struts Portal framework.

Basics elements of Struts

A Struts application is basically made up of the following elements:

Actions: The Actions represent processing that occurs prior to going to another page.

Pages: These are usually JSPs, but sometimes are HTML pages.

Action Form Beans: ActionForms are Beans associated with Actions, supplying the data to fill or collect various fields on a page.

The application writer creates these objects and, in the configuration file struts-config.xml, defines the relationships between these objects and the transitions that occur. The configuration of an ActionMapping associates a path with an

Action and with ActionForms, and can list one or more destination pages following execution of the Action. As a result of executing the Action, an ActionForward object is returned which contains the path to the next request. Typically, the returned path is to a page, but it is also possible that the path is to another Action. Figure 3 depicts the normal structure of a Struts application.

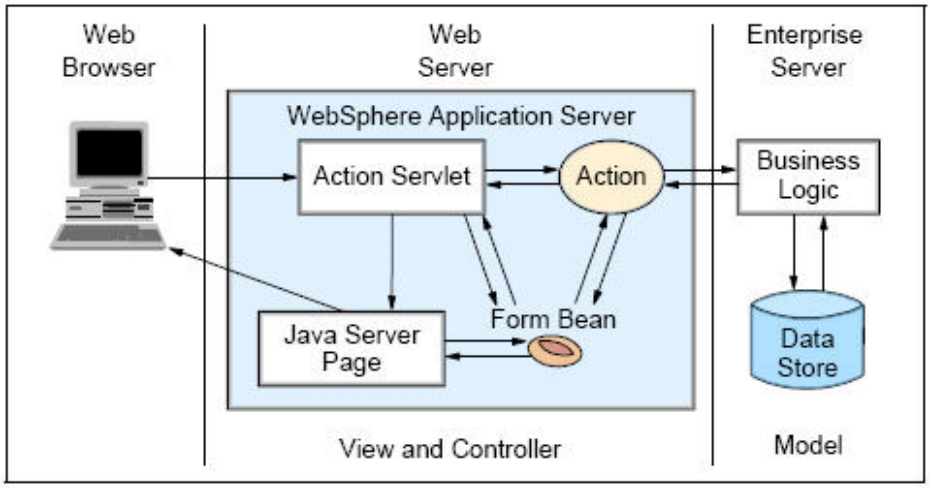

Figure 3 Struts structure

Struts 1.1

The design of Struts has evolved into a more modular implementation in version 1.1. There are two main new features in Struts 1.1:

The addition of the Request Processor.

The processing of a request has been moved out of the Action Servlet into the RequestProcessor class. The Request Processor is split into numerous process methods that can be overridden in a derived class to control the processing. The new modular implementation facilitates the support of Struts in the Portal server by allowing Request Processor methods to be extended as necessary. The

modular implementation limits the number of changes necessary to the base Struts itself.

Enhanced sub-application support.

Struts allows configurations that are prefixed, which allows different configurations based on prefixes. Portal server makes use of the prefixes to support Portal server's modes and device support.

Struts in a Portal server

A Struts application in Portal server is similar to the servlet-based Struts application. A WAR file that contains the Struts jars, the JSPs, actions, and configuration is built. The WAR file in Portal server has some additional requirements. There are some additional jar files, and a portlet descriptor file. There are also some necessary changes to the Web deployment descriptor and the Struts configuration.

To support existing Struts applications and offer a migration path to the Portal server for existing Struts applications, several differences between Struts servlet-based applications and Portlets exist, specifically:

- The portlet itself is a servlet and has its own processing and servlet mapping. The portlet servlet must be configured as the main servlet rather than the Struts Action Servlet.
- Portlet action processing (for example, handling of user events like selecting links or submitting forms) is done in a step prior to rendering the display view. During this processing the response object is not available.

- Display view rendering is done separately from action processing, and may be done multiple times without intervening actions. This occurs because all Portlets on a page are essentially asked to refresh or re-render themselves when the entire Portal server page is refreshed.
- URIs under Portal server has a special format. A special API is used to generate the URI and add the desired information to be passed to the portlet. If portlet URIs were not used, control would not get passed to the correct portlet. Thus, the portlet URIs must be used to get control passed to the correct portlet, with the additional path information needed by the Struts application made available. The Struts tags have been modified to automatically provide this needed functionality.
- Portal server does not support forwards or redirects. Either alternatives have to be used, or the functionality emulated. The differences enumerated result in the following conclusions with regard to supporting Struts applications as Portlets:
    - The Portlet's servlet mapping has to be used. Routing of Struts actions back to the Struts application processing has to be done using portlet URIs.
    - The processing of Struts Actions must occur during portlet action processing, and the information necessary to render a display view must be stored away so it is available during the later view rendering phase, which may occur multiple times without intervening actions.

- The Struts application paths, both to Actions and to pages, must be sent and retrieved in a different way.
- Forwards to pages (for example, JSPs) are done with include operations rather than with forward operations. Forwards to other Actions can be handled by recursively executing the Struts Action processing. Redirects are treated in the same way as a forward.

Two-phase processing

A portlet has a different processing and rendering design than a servlet. A servlet does all of its processing in the service method. A portlet, on the other hand, uses two-phase processing that is split between an action processing and service. The reason that a Portal server needs this split is because Portlets may need to communicate with other Portlets before the rendering stage. The action processing is guaranteed to complete before a portlet is asked for a rendering. The processing of events by a portlet is handled in a separate, earlier step from the rendering step. The typical action handling is achieved by implementing the ActionListener interface. The ActionListener interface provides the actionPerformed method, to which an ActionEvent object is passed. When a user clicks on a link or a submit button, an ActionEvent can be generated. The PortletAction can be obtained from the ActionEvent, which describes the triggering event. When the actionPerformed method is invoked, a response object is not available because this is not a rendering step.

The rendering step is handled in a separate method, the service method, which calls the doView method in the PortletAdapter. The doView method is not only

called following the actionPerformed processing when a user clicks on a link or button in a portlet, but is also called when the portal page is refreshed. Thus, given a page with two Portlets, A and B, when the user clicks on a link in portlet A, actionPerformed and doView is called for portlet A, but only the doView method is called for portlet B. The two-phase processing causes some significant challenges as it relates to Struts applications:

- The methods called during Struts processing expect both a request and response to work with. The first issue of the lack of a response object during actionPerformed processing can be handled by creating a pseudo response. The response object used prior to actually invoking the JSP is not heavily used. However, during action processing within a Struts application, it is not unusual for an application to report an error using the sendError method.
- The Struts rendering of the page is usually immediately preceded by action processing; they are essentially parts of one step. The second issue is the fact that the rendering step is separate from the event handling step. Note that some of the information that was available during the processing of the actionPerformed method, namely the request parameters, is no longer available when the doView method is invoked. Additionally, since doView can be called when the portlet page is refreshed (without a new event occurring for that portlet), all information required to render the page must be available every time doView is called.

Since normal Struts processing allows the updating of the response object during almost all steps, including during action processing, there are clearly situations where the two-phase model would cause problems in the application. However, a typical well-behaved Struts application does not write directly to the response object and instead forwards to a JSP. It is assumed that the JSPs do not themselves need access to the original event information, request parameters, which initially led to the current rendering. Instead, it is assumed that the Action processing stores information in a bean, from which the JSP extracts the information. As long as those beans are available during the rendering, the rendering can be repeated. Since those beans, most notably the ActionForm, are usually stored in the original request attributes, the request attributes from the original request processing must be saved and made available when doView is invoked.

No response object

The typical well-behaved Struts application has no need to access the response object during the Action processing. The lone exception encountered is the need to report an error through the sendError method. When an application checks and finds some state information invalid, it is common to use response.sendError to generate an error page. Failing to support this would break a large number of legacy Struts applications. The Struts portlet framework intercepts the calls to sendError and saves the error information for display at a later time. During the later view rendering phase, this error information is found, and the error information displayed instead of displaying other content for the portlet.

The Struts URL paths

Struts Action mappings are defined in terms of paths. The name and location of page objects (for example, JSPs) are defined via paths as well. Thus, although Portlets have their own form of URI, it is still necessary to be able to associate the Struts path with an action sent to a portlet and to retrieve that Struts path when the portlet action is handled. It is not unusual to pass parameters on such a path via the query string on the HTTP URL. It is also important to realize that often the actions containing these paths are generated from tags provided by Struts. The most obvious examples of these are the tags for the HTML elements LINK and FORM. Obviously, in order to support such tags when the Struts Application is executed in a portlet, they have to be modified to create a portletURI with the required additional information. The Struts tags that create links have been modified to pass the created paths directly through to the Portal server as a parameter on the portlet URI. This allows a portlet URI to be created that will cause the interaction with the portlet.

Forwards and redirects

As mentioned earlier, the Struts Action objects return an ActionForward object, which contains a path to be either forwarded or redirected to. Portal server does not support forward operations because the response object has already been committed. If the path is not an action, the page is included instead. If the path is an action, then a forward to the Action is simulated. To include a page use the PortletContext include method.

A forward for an Action is simulated by recursively sending the new Action URI through the Struts base processing of an Action. Note that not only can we have this issue of forwards in ActionForward objects returned from Action objects, but also in tag handlers. In tag handlers, it is possible to access the PageContext object and invoke the forward method as well. An alternative method to the PageContext.forward is available via the PortletApiUtils class, which provides a mechanism to emulate the forward functionality. It is important to understand that although we can write Portlets using Struts, there are some things that we can do in a Struts servlet that we cannot do in a Struts portlet. For example, Struts provides support for things like redirects and forwards to other servlets. These are provided because they are functions generally available to servlets. However, these capabilities are not available in Portlets.

What not to do in an action

Following are two examples of what we shouldn't do in an action:

The typical use of the response object is to call sendError when an error condition is detected. Portal server does not support writing to the response object during the action processing. Therefore, a response object is not available. A pseudo response object is made available during the processing in the Request Processor. If the Struts Action writes to this response object, that text will be lost. The Action should return an ActionForward object. This is important so the Request Processor goes through the normal processing.

The ForwardAction and IncludeAction, normally shipped with Struts, are examples of what should not be done in a Struts application in Portal server. These actions create a RequestDispatcher and try to forward or include the path in the Action execute. The ForwardAction should return an ActionForward so the RequestProcessor can process the Action. The Struts portlet framework provides its own versions of ForwardAction and IncludeAction to deal with this problem.

Roles support

Struts use roles to determine access to an Action. Portal server does not support roles at this time. The Struts portlet can be configured to use group names as if they were role names, in addition to group names, to emulate role-based access control. The following init parameter in the Web deployment descriptor enables this support:

```
<init-param>
        <param-name> UseGroupsForAccess</param-name>
        <param-value>true</param-value>
</init-param>
```

Chapter 5: Portlet Development Guidelines

The Portlets

When we implement one portal solution, it is possible to integrate information from multiple sources in our company in a single point of access. In this case, we have one infrastructure to support different types of devices and data sources. As we begin the process of planning and implementing a portal, keeping the following guidelines in mind:

- Create components with object-oriented design principles. Design Portlet applications according to good object-oriented design principles with loosely coupled, encapsulated components to maximize the opportunity for reuse.
- Avoid writing Java code in JSP. Use Java beans to encapsulate the interface operation.
- Understand the customer's problem. Take the time to fully understand the requirements of a comprehensive user profile to support all related applications before starting the application design process.
- Understand the Portal solution concept, and use the Portal in the way it was intended: as a framework for aggregating application components. Design JSPs, servlets, beans, and so forth, to handle all of the business processing and plug into the portal framework, rather than developing Portlets. This will maximize flexibility of the solution and increase platform independence.
- Avoid making default Portlets user-customizable; create a default page using custom Portlets and persistence based on a key comprising the

page, portlet, and user ID to allow for a default page. Do not use the default persistence model for shared Portlets since changes made by one user will be seen by all.

- Avoid using public sessions, but instead write Portlets to be as session-independent as possible, at least when the user is anonymous.
- A clear definition of success and measurement criteria at the beginning of a portal effort is critical. Clear definitions of the goals, functions, and measurable success criteria will help lead to a successful effort.
- Existing business rules must be evaluated and potentially changed to fit into the user experience.
- Consumer trend data must be constantly monitored and acquired, internally and externally, to be useful. Applications must be developed with flexibility.

The vast amounts of data that can be collected must be managed and transformed to be worthwhile. This solution should cover much functionality and provide flexibility for the user. The Portal Solution provides some concepts to help us achieve this level of quality. These concepts, such as Single Sign-On, personalization, and internationalization, are ones that the developer should take into consideration when developing the solution.

Portlet coding guidelines

These guidelines are intended to help produce best-of-breed Portlets for the WebSphere Portal environment. Since Portlets are very similar to servlets, it is a

good practice to follow the same performance tips that apply to servlets and JSPs, such as connection pooling, prepared statements, and so forth.

In addition, we should consider the following coding guidelines:

Use no instance and class variables

The most important rule of thumb is to limit the use of instance or class variables in a portlet. Within WebSphere Portal Server, only a single instance of the Portlet class is instantiated. Even if different users have the same portlet on their pages, it is the same portlet object instance that generates the markup. Therefore, each portlet instance has to be treated like a singleton. The implication is that we should not use instance or class variables to store any state or other information between calls to the portlet. Being a singleton, each instance variable of a portlet effectively becomes a class variable with similar scope and the potential for problems when the variable is accessed many times in each user's context of the portlet. It is better to avoid these problems by limiting the use of instance or class variables to read-only values. Otherwise, re-entrance or thread safety may be compromised. There is an exception to this rule in which we can use instance variables. Variables whose values do not change can be stored safely in instance variables. For example, on initialization of the portlet, its configuration is set within the portlet. Because the configuration does not change for different users or between calls, we can store it in an instance variable.

Use try/catch blocks

Get in the habit of adding try/catch blocks around significant code in JSPs to enable debugging. This will greatly improve our portlet development experience.

Minimize dependencies on JavaScript

Since JavaScript implementations and behavior differ widely between browser types and versions, the more our portlet depends on JavaScript, the more browser-dependent our portlet becomes. Additionally, the more of the page that is rendered using JavaScript, the more difficult it is to maintain and to extend to other markups besides HTML. Also, it is more difficult to namespace-encode JavaScript resources and nearly impossible to properly encode (response.encodeUrl()) URLs built using JavaScript.

Use taglibs whenever possible

Encapsulating Java code within taglibs not only allows common view functions to be easily reused, it keeps the JSPs clean and makes them more like normal HTML pages, which allows the page designer to concentrate on layout and decoration and reduces the possibility of breaking the embedded Java code.

Pass data to the view (JSP) as a bean in the request object

Use the PortletRequest object to pass data to the view for rendering. This way, when the request is complete, the data falls out of scope and is cleaned up. Passing it as a bean allows the JSP to simply refer to the data as properties on the bean using intrinsic functions in the JSP syntax.

Use the portlet logging facility

Using the PortletTraceLogger installed with WebSphere Portal for tracing and logging allows our Portlet's debug and trace output to be stored with other Portlet output in the Portal log files. It also takes care of the infrastructure around logging

and tracing. The PortletLog object that is the interface to the logging mechanism can be obtained from the Portlet context:

PortletLog log = getPortletConfig().getContext().getLog();

We should also verify that logging is enabled for attempting to log data, especially if that requires additional logic or storage to build the log message.

Example 7 Using a log from portlet

```
if (log.isDebugEnabled()) {
String logMsg = new String("Time to retrieve data: " + elapsedTime);
Log.debug(logMsg);
}
```

Adopt good code documentation habits

While commenting of code is not required for the functionality of the portlet, it is essential for its maintenance. In addition to the fact that the responsibility for a Portlet's maintenance can change hands over time, well-written Portlets serve as models for other Portlets, which means that someone else must understand what we wrote. Well-documented code implies more than simply inserting comments; it implies good naming practices for Java resources as well. The following are examples of guidelines to follow:

- Insert JavaDoc-compliant prologues for all public classes, variables, and methods. Be sure to document inputs and outputs.
- Include inline comments, but do not include them on the same line as Java source. It is difficult to align and follow comments which are on the same line as Java source.
- Use meaningful names for variables and methods. Variables x, y, and z, while easy to type, are not easy to follow through code. Capitalization conventions

and the use of underscores ('_') within resource names is a matter of personal choice, but be consistent once the choice is made.

Use the ContentAccessService to fetch external content when necessary

If it is necessary to fetch external content using an HTTP request, use the ContentAccessService as it is intended to perform that function as expediently as possible. It also prevents us from having to rewrite the same function and introduce another maintenance point.

Cache portlet settings or portlet data

If portlet settings or data is complicated and requires complex parsing to digest, it may be a good idea to cache the data for quick retrieval later. Avoid using the PortletSession to store this data as it causes "session bloat." Alternatively, consider using the Application Server's Dynacache CommandCache to store this data. PortletSettings can be stored keyed off the Portlet's name. PortletData will have to be stored keyed off the portlet name and user name. When updates to the cache are necessary, based on updates to PortletData or PortletSettings, devise a means of communicating updates to the data, such as through implementing the PortletSettingsAttributesListener interface, or by using a "dirty bit" that can be interrogated to see if updates have been made to the configuration, in which case the cache can be invalidated and updated.

Note that the CommandCache is not currently cluster-aware, meaning that cached content is not shared across horizontal nodes in a cluster. So, portlets must be able to create the cache entry from scratch if the cache entry does not exist.

Follow design guidelines for Struts Portlets

If we are to develop Portlets based on Struts, use the following additional guidelines to ensure our implementation is the best it can be:

- Use the sample portlet applications that accompany the Struts Portal Framework package to ensure Struts is working properly in our environment before testing Struts support in the portlet.
- Review the documentation on the Struts Portal Framework for application requirements and any restrictions. For existing Struts applications refer to the section in the same document titled "Migrating an Existing Struts Application."

Portlet packaging guidelines

Make common functions available externally to Portlets

If the portlet contains common functions that are replicated across several Portlets, consider isolating them and making them externally accessible by the Portlets. The easiest way to do this is build another JAR file of the common classes and place the JAR file in a location that is in each Portlet's classpath, such as the AppServer/lib/app directory. Another approach is to build a Portlet Service from the common functions, which can then be accessed using the common PortletService interface and retrieved using the PortletContext.getService() method.

Combine Portlets with similar functions into a single portlet with multiple configuration settings

If we are developing several similar Portlets, such as a mortgage calculator portlet, car loan calculator portlet, and a home equity loan calculator, it might make sense to actually develop a single portlet that can behave differently based on configuration settings. The common portlet code is specified as part of the portlet application's abstract portlet definition. The application's various concrete portlet definitions have unique configuration parameters (PortletSettings) that can be used to customize the behavior of the common portlet code.

Data management

There are three primary areas where portlet configuration information is stored and can be maintained: portlet settings, portlet data, and servlet configuration.

Use portlet settings to store user-independent configuration data

This data represents configuration parameters from the Portlet's portlet.xml file (<config-param> elements under the concrete portlet definitions). It is only writable in the portlet's configure mode (doConfigure() method), but is readable through the getPortletSettings() method on the portlet.

Use portlet data to store user-dependent configuration data

This data often represents personalized overrides to the portlet settings and is thus stored according to user and portlet instance. This data is only writable from a portlet's edit mode (doEdit() method), but is readable through the getPortletData() method on the PortletRequest object.

Use servlet configuration for storing static initialization information for a portlet

Examples of this information include the install directory or path information. This data, taken from the portlet's web.xml file, is read-only and accessible using the getServletConfig() method on the portlet.

General session management guidelines

Limit the use of the portlet session for storing portlet state information

The Portlet Session is a convenient place to store global data that is user-portlet-specific and that spans portlet requests. However, there is considerable overhead in managing the session, both in CPU cycles as well as heap consumption. Since sessions may not expire for quite some time, the data contained in the sessions will remain resident in active memory even after the user is done with the portlet. We should be very judicious about what is stored in the Portlet session. For example, parsed configuration data (from PortletSettings) should not be stored in the Portlet session since it can be recreated from the database at any time.

Do not rely on portlet sessions if the portlet is to allow anonymous access

If our portlet is to be used by unauthenticated users, such as on the Welcome page, then the portlet should not rely on the portlet session at all. Portlet sessions are bound to a user context. By default, for unauthenticated users, a portlet session can be requested by a portlet, but it is a temporary session which only lasts for the duration of the current request. Public session support exists,

where sessions are issued to unauthenticated users, but because there is no log out action for an unauthenticated user, the session will not be invalidated until it times out, which can lead to severe memory consumption issues. There are other ways to preserve global data for anonymous access, such as:

- Storing data in a collection whose key is communicated back to the client using a cookie.
- Storing data in the WebSphere Application Server's dynacache facility, again using a unique key which is communicated to the browser using a cookie.
- Storing the state information as a cookie itself.
- Storing non-sensitive data as hidden field elements within Forms so that it gets passed back on subsequent form submission. This has the added advantage of binding state-aware data to the page requiring it.

Alternatively, consider limiting the function of the portlet for anonymous access.
We can check for the presence of a user object (PortletRequest.getUser()) to see if a user has logged on to the portal, in which case a portlet session should be available.

Always request an existing portlet session

Always request a portlet session using either PortletRequest.getPortletSession() or PortletRequest.getPortletSession(false), which will only return a session if one does not already exist. The portal server will establish a portlet session for a portlet before it is invoked. This helps prevent the case where a temporary session is generated for a portlet during anonymous (unauthenticated) access.

Prevent temporary sessions from being generated in the JSP

Add the JSP page directive

<%@ page session="false" %>

to the JSP to prevent temporary sessions from being created by the JSP compiler if none already exist. This will help guard against attempting to use the session to store global data if the session will not exist past the current request. We will need to be sure the portletSession exists before trying to use it.

Other general development guidelines

There are potentially many different types of browsers, or user agents, which access WebSphere Portal besides a typical desktop browser, such as hand-held devices (PDAs, or personal data assistants) and wireless phones. The capabilities of these devices vary widely, as do their users' attention spans and interaction models. Therefore, the Portlet's design needs to take these alternate devices and users into consideration.

Information priority is different depending on the device it is viewed on. Capabilities of desktop browsers are limitless. Users can download almost anything quickly and randomly navigate through it. For handheld devices, however, screen real estate and device memory is at a premium, not to mention the relatively slow speed of a wireless connection versus a LAN connection. Also, users of mobile and handheld devices need quick access to concise information and cannot afford the time or effort it takes to navigate through several screens to get to the desired information. With these limitations in mind, here are some more general guidelines for portlet development:

All portlet strings should be fetched from resource bundles

All displayable strings should be stored in resource bundles and fetched using the ResourceBundle Class. The resource bundles should be organized by language under the Portlet's WEB-INF/classes directory. For example, if a portlet supports English and Finnish Language, it would have three resource bundles (default, English, and Finnish) and might be organized under WEB-INF/classes as:

WEB-INF/classes/nls/mystrings.properties
WEB-INF/classes/nls/mystrings_en.properties
WEB-INF/classes/nls/mystrings_fi.properties

Using the ResourceBundle class, we would refer to the properties file as "nls.mystrings". Java will automatically look for the appropriate properties file based on the current locale.

Organize portlet help files by language

As opposed to typical JSPs in the Portlet's view, the Portlet's help JSP files can and should be language dependent. Since little else in the JSP will exist but help content, there isn't much need for extracting strings from resource bundles. Translate the JSPs instead, and organize them by language. When using the include () method on the portlet context to dispatch a JSP, the method will search the directory structure for locale-specific subdirectories first. For example, if we include the JSP "*/WEB-INF/helpjsps/html/myHelp.jsp*", and the current request locale is en_US, then the method will look for the myView.jsp in the following directories, in order:

/WEB-INF/helpjsps/html/en_US/myHelp.jsp

/WEB-INF/helpjsps/html/en/myHelp.jsp
/WEB-INF/helpjsps/html/myHelp.jsp
/WEB-INF/helpjsps/myHelp.jsp

Note that the include () method will also look for a default myHelp.jsp if one does not exist in the HTML subdirectory.

Use portlet messaging to communicate to other portlets on the same page

It is possible to send messages to a specific portlet or to all Portlets on a page.

The message body may contain data that the receiving Portlets require. The message is sent from a portlet's actionPerformed() method and received by other Portlet's messageReceived() methods, before those Portlet's doView() is called as a result of the action, so there is an opportunity to set data or state information before the receiving Portlets are invoked.

Avoid the use of HttpSession to share data with other portlets/servlets

The HttpSession is not common between nodes in a cluster, so any data store in the HttpSession is only visible by J2EE resources in the same Application Server instance. The PortletSession is derived from the HttpSession, and besides only having visibility of the portlet instance, is also bound by the same restriction. Persistent sessions can be turned on in the application server, which will serialize the session to a database which can be shared across nodes in a cluster, but that can cause performance issues for very large sessions.

Use Internationalization and National Language Support (NLS)

To reach as many users as possible, WebSphere Portal supports different languages for different locations. For instance, a large, international corporation

may address users in different countries or regions through multilingual Web sites. In this context the portal can concurrently serve portal views to large numbers of users, each in the user's preferred language. WebSphere Portal V5 supports, by default, 24 different languages. We can add language support to our Portlets by creating different JSPs that contain different languages. These JSPs are then packaged in the correct directory structure as described. In Example 8 we can see two JSPs supporting different locales.

Example 8 Locale support

```
/jsp/html/ie5/en_US/view.jsp
/jsp/html/ie5/da_DK/view.jsp
/jsp/html/ie5/view.jsp
```

Use device-independent development

When designing and developing portlet applications; it is always a good idea to use a device-independent development method. This does not mean that all our portlet applications should support multiple devices (which would be a waste of time and money), but instead, it means using design and development methods that will enable our application to be easily extended to support multiple devices when necessary. This can be done simple by using the different functionality leveraged by WebSphere Portal V5.

There are two fundamental methods for providing device-specific content:

- Generating the content in the code in a programmatic way. It is a direct solution using either different presentation pages or style sheets.
- Using an intermediary component, like a transcoding engine, that takes care of the content transformation. It is a configurable, intelligent engine

that is capable of recognizing the device and transforming the content as required on the fly.

WebSphere Portal provides several ways to support the diverse characteristics of the client. For example, we can use the getMarkupName() method to check the markup supported by the client. However, the portal aggregation component allows us to package JSPs and resources in directories that match the client with the request, reducing the amount of programming we have to do to support multiple client types. JSPs that contain mostly text, such as help JSPs, can be translated directly rather than storing strings in a resource bundle. JSPs that do not use resource bundles need to be stored in the appropriate location. When a portlet uses a JSP for rendering of the Portlet's content, the portal searches for and selects the proper JSP based on the client type (including browser), markup language, and locale indicated in the request. To include a JSP in a portlet, we must use the function PortletContext.include():

Example 9 Using PortletContext.include()

```
public class DevicePortlet extends PortletAdapter {
    public void doView(PortletRequest request, PortletResponse response)
        throws PortletException, IOException {
            getPortletConfig().getContext().include("/jsp/view.jsp", request, response);
        }
}
```

To support multiple markup types and locales, the Portlet's JSPs must be packaged in the WAR file using the following directory structure:

jsp_path/markup_type/language _region/client/jspname.jsp

where:

jsp_path is a path defined by the developer. For example, JSPs can be located in the root of the WAR file or in a JSP directory. However, this path must not

include mime-type/language_region_variant. The include() method already locates the correct JSP, also in those directories.

markup_type is either html, wml, chtml, or any other markup language we have defined in our portal.

language is the language for this JSP, for example, en, ja, or de.

region is the country, region, or territory of the JSP, for example, US, UK, or CA.

client is the type of device. For example, it could indicate a JSP with browser-specific markup, such as Internet Explorer or NetScape 4. Manage Clients help describes how clients are identified by the portal server. For example, if the client is using Internet Explorer 5 with language properties are set to English (United States), the method: *.include(/jsp/view.jsp, portletRequest, portletResponse)* would cause the portal server to look for the JSP in the following order:

/jsp/html/ie5/en_US/view.jsp
/jsp/html/ie5/en/view.jsp
/jsp/html/ie5/view.jsp
/jsp/html/en_US/view.jsp
/jsp/html/en/view.jsp
/jsp/html/view.jsp
/jsp/view.jsp

Content accessed by a portlet JSP (for example, images or HTML pages) using ServletResponse.encodeURL() are not found by portal aggregation. To provide different versions of these resources based on client type, markup, and locale, the portlet must use PortletContext.include().

Performance considerations

This section focuses on performance analysis and guidelines for the design of our portlet applications. It also introduces examples and provides simple guidelines for tracing response times and for debugging applications from a performance standpoint, specifically:

- Timers
- Response time tracing
- Threads
- Caching
- Garbage collection
- Object pooling

Performance analysis

An e-business solution often differs from a traditional solution by having several functional nodes mapped to several products and custom-built applications.

From the time a client makes a request until the client receives the response, many different components have to be invoked. Having many different and individual components always introduces numerous potential bottlenecks to the solution. When architecting a solution today, a vital part of the job is to ensure our solution will perform in accordance with the non-functional requirements. The non-functional requirements should be included in our test cycle as well.

Sizing of the operational environment to support the running software and applications is just as important as having optimized the custom-built applications.

Timers: Having multiple Portlets on the same portal page can often make it difficult to track response times of the different portlet applications. For that reason, putting timers in our application code is always a good idea. That way we can track the complete response time of a particular application to see if it is performing as expected. Implementing timers around critical operations also gives us a strong tool for finding bottlenecks, like calling third party APIs or enterprise backend that we are dependent on. This can save us a lot of time when trying to identify performance problems and issues. It is very easy to implement simple timers in the application code. Example 10 shows how simple it can be.

Example 10 Simple timer

```
public String getAccountName() {
    long begin = System.currentTimeMillis();
    try {
        // .. do operations
      } catch ( Exception exception ) {
        // .. catch exceptions
      } finally {
        long delta = System.currentTimeMillis() – begin;
        System.out.println("Timer: " + getClass().getName().getAccountName() + " took ms: " +
delta );
      }
}
```

This code is a very simple timer that uses the System to get the current time and to print out the time difference. Depending on our operational environment, writing to the file system is not always a good idea, but this example is, of course, intended for development purposes only. After a development and unit test cycle

has been completed, our application is often moved to a staging environment that resembles the real production environment. Making a final performance test on our application running in an environment that mimics our actual production system is very important, especially when our applications are integrating against third party products, enterprise backend, and so forth. Think of implementing timers as logging, like with the open source logging frameworks log4j or jLog, and try to make it a natural thing to do when logging. We can implement a timer switch to enable and disable timer logging.

```
if(Timer.isTimerEnabled())
System.out.println("Time is ....");
```

We can also have our timers logged to a database instead of the file system.
This is a good approach since it enables us to run performance validation on production systems without a major overhead. In this way we can verify our system is performing as required according to our specifications, and as appropriate, repeat the scenarios as the system improves or expands.

Response time tracing: It is very easy to trace our response time from the client making a request to receiving a response back. This can be done by various small open source programs or by IBM Page Detailer. But unfortunately, it cannot show which portlet used how many milliseconds, or if we coincidently had garbage collection that made our response times peak. To get the bigger picture on which components takes up our response times, we first need to identify the information sources. The following are some of the possible sources:

- Network
- Firewall

- Network dispatcher
- Authentication proxy
- HTTP server
- Application server
- LDAP server
- Content management systems
- Portal server
- Portlet applications
- Database
- Domino/Sametime/Quickplace servers
- External applications

When optimizing for performance, tracing and logging is very important. Therefore, enabling as much logging as possible in regard to response time is crucial. For example, enabling the response time logger within IBM HTTP Server is often a good starting point. As an example, comparing the response times in the HTTP server's access log to the client's response time log can often give us a good idea of where a bottleneck exists, either in front of or behind the HTTP server. This information will help us decide where to make adjustments.

It is important that we time our applications in order to pinpoint bottlenecks, especially if we are integrating with external applications over which we have no functional control. Monitoring our database is another important step in understanding the performance picture. To do this:

- Collect resource consumption data (cpu utilization, memory allocation, paging, and so forth).
- Log SQL statements executed Run each of the SQL statements through an explain tool to get the indexes suggested, and then compare if they already have been applied to the database.

These examples identify just a few of the important places to look for and enable logging. When we have collected all the data necessary we can start comparing it. A very efficient way of doing this is to import the data into some kind of spreadsheet. Statistical analysis based on the data will show us how our different system components performed at a specific time. We can, for example, see that when we had a 2 second response we also had a garbage collection, or that the backend server was paging. Now we actually have a concrete tool for pinpointing bottlenecks and targeting our optimizing efforts. For example, should we change garbage collecting policy or increase the memory on the backend system?

Garbage collection: Our garbage collection policy is always an important factor when tracing for performance. The garbage collection implementation is key to the superior performance of the IBM Java Virtual Machine (JVM). While most other JVMs need a lot of tweaking to provide optimal performance, IBM JVMs are designed to work well in most scenarios with their “out of the box” settings. In certain situations, however, garbage collection performance goes down for no apparent reason. The result can be an unresponsive server, a frozen screen, or a complete failure.

When a Java thread makes a request for a piece of storage, and the JVM is unable to locate a large enough chunk of storage to satisfy the request, an allocation failure (AF) is said to have occurred. This is when garbage collection becomes unavoidable. Garbage collection involves collecting all the "unreachable" references so that the space consumed by them can be reused. It is carried out by the thread that makes an allocation request, and is a Stop-The-World (STW) mechanism; all other threads of the Java application are suspended while garbage collection is going on.

On a multiprocessor system with N CPUs, a JVM supporting parallel mode starts N-1 garbage collection helper threads at the time of initialization. These threads remain idle at all times when the application code is running; they are called into play only when garbage collection is active. For a particular phase, work is divided between the thread driving the garbage collection and the helper threads, making a total of N threads running in parallel on an N-CPU machine. The only way to disable the parallel mode is to use the **-Xgcthreads** parameter to change the number of garbage collection helper threads being started. When using concurrent mode, a background thread is started by the JVM (independently from the garbage collection helper threads), and some part of the work is done in the background while the application threads are active. The background thread tries to complete part of the garbage collection work concurrently with the application, so that when garbage collection does kick in, the resulting STW pause will be reduced.

In some cases, though, concurrent processing can have a negative impact on performance, especially for CPU-intensive applications. Use the -**verbosegc** command line parameter to track our garbage collection operations. Enabling verbosegc can have a performance impact on an application. If such impact is unacceptable, a test system should be used to collect the verbosegc logs. It is common for server applications to keep verbosegc enabled at all times. This can be a good way to monitor the overall health of the JVM, and can be invaluable in post-mortem if an OutOfMemory error occurs.

Performance guidelines

In this section we present a conceptual overview on caching and object pooling strategies. Both can help to build scalable and well performing portlet applications. In addition, we point out some common performance pitfalls in portlet coding.

Caching: Caching is a vital part of any Web application when we are trying to optimize performance, but caching can also be very difficult to implement. As solution and application designer, we should consider the caching possibilities as early as possible in our designs. There are two important forms of caching:

Product-delivered caching

> The standard products that are typically used in Web applications often offer different caching mechanisms that can be used. It is important to understand which products in the operational model offer caching, and how they can be combined.

HTTP servers

HTTP servers typically offer caching of static content, like images, static html files, JavaScript, and CSS files. It is generally a good idea to have the HTTP server serve these items instead of the application server.

Application servers

Application servers like WebSphere Application Server have servlet caching for caching content delivered by the Web applications running at the server. They also deliver prepared statement caching for optimizing access towards the used database.

Portal server

WebSphere Portal has different cache settings, which we can tune in order to gain performance improvements. These cache settings are stored in property files in the WP_ROOT/shared/app/config/ directory. There are no particular settings to gain the most performance; instead, the settings have to be in sync with the non-functional requirements of our solution.

As an example, we should consider how long our impersonalized pages are to be cached. By default, WebSphere Portal only caches them for 60 seconds. If we are not creating, updating, or deleting pages several times an hour on our production system, we might want to consider increasing this setting. On the other hand, we do not want to wait hours to see a change take effect.

Another example is the caching of the user's access rights. Often the user's access rights are based on the group of which he is a member. Consider the time that access rights are to be cached, based on the

relative stability or volatility of our access group memberships and individual access rights changes. It is a fine line to find the settings that match exactly our portal solution and it often takes a couple of tuning iterations to find the optimum values.

Databases

When using the statement caching possibilities leveraged by an application server like WebSphere Application Server, we also have to match these settings on the database as well. Having the statement caches and connection pools for our different databases, along with the max applications, in sync with the application server is important to avoid bottlenecks.

Authentication proxies and directory servers

Authentication proxies like Tivoli Access Managers WebSeal can cache static content along with user credentials retrieved from, for example, IBM Directory Server. LDAP operations are in some cases considered expensive, therefore caching is very important. As these examples illustrate, there are many different caching options within a typical operational model. This is a vital operation; the products do not pre-calculate our caching needs for us!

Programmatically delivered caching: This is another important caching mechanism is the one can be implemented in our application code. When retrieving data from data sources; always consider whether this information is cacheable. Is it a light or heavy data retrieval process? What type of information

is it? For example, caching user preferences is a good idea, but caching values of stock options for several hours might not be such a good idea.

If portlet output is static in nature or is valid for some period of time before it is updated, the portlet should enable the "display cache" by setting appropriate values for the <cache> elements in the Portlet's portlet.xml file (note that caching must be enabled in a portlet before it is installed or updated). A Portlet's caching settings can be set such that the output is never cached (<expires> value of 0, the default), cached for some period of time (<expires> value greater than 0, in seconds), or cached indefinitely (<expires> value of –1). The cached entries can also be shared across all users of the same portlet (<shared> value of YES) if the output is valid for any user. Using the cache vastly improves the performance of the portlet, and thus of the page in general. The portlet can also implement the getLastModified() method to control the invalidation of the cached content, informing the portal which it should call the Portlet's doView() method to refresh the output.

Object pooling: Pooling objects like sessions to backend systems can help to significantly improve performance in a Portal environment. In a typical workplace we usually have more than one Portlet on a page. In addition, it is usual for many users to have access to the same content. These circumstances can lead to limited-resource problems. Limited resources can cause performance bottlenecks when there are not enough resources to meet clients' demands. For example, sessions to backend servers require non-trivial amounts of time to create and destroy. Often, in high-throughput scenarios, Portlets must wait for a

session object to become available, creating a bottleneck in the flow of the application. With object pooling, many Portlets can share a limited resource such as a session, using it only when they need it. In this way, the performance cost of creating and destroying the session is reduced. Implementing an Object Pool can provide many advantages since the Object Pool:

Is more scalable, resourceful

We can share objects between multiple Portlets and reuse objects for users in the same user group.

Enables load balancing

A pool manager can manage sessions to multiple backend servers at one time and allow load balancing and fail-over.

Is thread safe

The pool manager is completely thread safe, meaning that it can be accessed safely from any number of threads—a situation that is common for Portlets.

Helps with session time-out

Session time-out occurs when a session is idle longer than the session time-out setting. After this time the session and all its associated objects become invalid.

Is easy to use

We do not have to consider the possible state of an object; we simply use the object. For example, if we want to retrieve data from a database; we do not need to know whether a session and database connection has been previously established, we can just retrieve our data.

Object pool: An object pool is a set of limited resources, such as sessions, that can be reserved for use by clients and then returned to the pool (for probable reuse) when the object is no longer needed. Reserving and returning pooled objects avoids the overhead of separately creating and destroying an object each time a client requests it. Multiple object pools can be used. For example, one object pool might contain session connection objects, and another pool might contain database or view objects.

Virtual and physical objects: Without object pooling, whenever a Portlet requests an object, a physical object is created, then the object is destroyed when no longer needed. By contrast, when a Portlet uses object pooling, the request for a pool able object generates a virtual object instead. The virtual object supports all the methods of the requested object, but the Portlet sees only the virtual object. When a Portlet calls an interface method from the virtual object, the virtual object's implementation requests a physical object from the pool and delegates the request to the physical object. When the request is complete, the Portlet returns the physical object to the Object Pool Manager for use by other virtual objects.

Client: In the context of object pooling, a client is the Portlet code that calls the Pool Manager. The Portlet requests objects, and the Object Pool Manager fulfills the requests or relays an error back to the calling Portlet.

Object Pool Manager: The Object Pool Manager is a service used by one or more Portlets. In response to clients' requests for objects, the Object Pool Manager controls one or more pools (for example, sessions) by reserving and

releasing the objects in the pool. The Object Pool Manager performs the following tasks:

- Queues virtual objects' requests for physical objects
- Marks physical objects as either free or reserved
- Attempts to create physical objects when necessary
- Destroys physical objects in a pool, based on idle time or usage limits

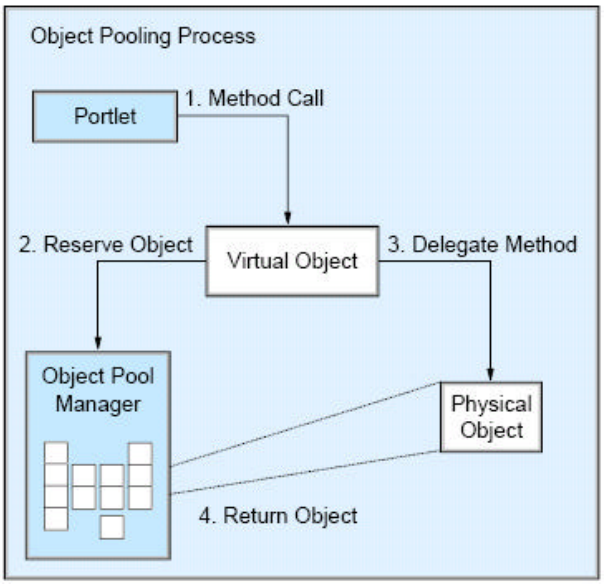

Figure 5 Object pooling process

Object pooling process: The object pooling process is as follows:

- The Portlet calls an interface method on the virtual object.
- The virtual object's implementation reserves a matching physical object from a named pool.
- The method call is delegated to the physical object.
- When the method call is completed, the physical object is returned to the appropriate pool for use by other virtual objects.

- The Object Pool Manager uses a timer thread that periodically releases unused physical objects after a time-out.

How the Object Pool Manager works: As illustrated in Figure 6, when a Portlet makes a request to reserve an object, the Object Pool Manager processes the request through the following possible phases:

- Matching a pooled object
- Creating a pooled object
- Replacing a pooled object
- Queuing a request
- Returning a FAILURE state

Details about these phases follow the figure.

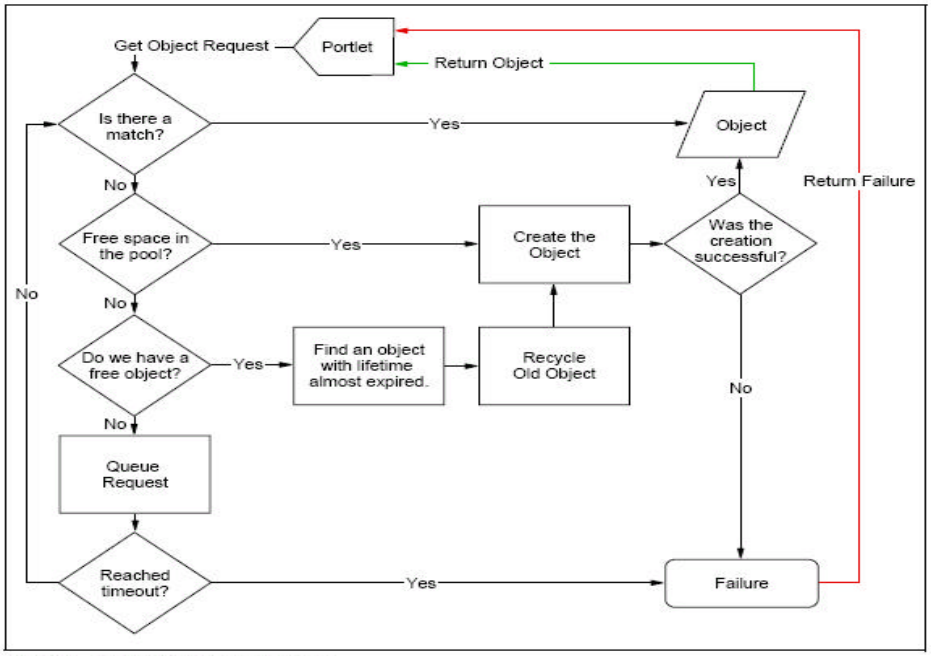

Figure 6 Pool Manager process flow

Matching a pooled object: A client makes a request of a virtual object. It communicates with the Object Pool Manager to bind the virtual object to a

matching physical object. The virtual object then repeats the client's request, making this same request of the matched physical object. If a match is found, the Object Pool Manager reserves the matching physical object and sends it back to the requesting client. If no matching object is found, then the Object Pool Manager proceeds to the next phase of processing logic—attempting to create a physical object.

Creating a pooled object: If no matching object is found, the Object Pool Manager checks its MaxPoolSize setting before trying to create a physical object. MaxPoolSize determines the maximum number of objects a pool can contain.

If the number of objects in the pool is less than MaxPoolSize, then the pool is allowed to grow. As a result, the Object Pool Manager calls the CreateObject( ) method. A successful call returns a physical object to the requesting client. An unsuccessful call indicates that a critical resource is unavailable. For example, if an application requests a session but the backend server is down, then the call to CreateObject( ) would fail, and a FAILURE state would be returned immediately.

Replacing a pooled object: If an object pool reaches the limit set by MaxPoolSize, the pool is considered full. As a result, instead of trying to create an object, the Object Pool Manager attempts to recycle an almost expired object and create a new one that matches the one requested. Only unreserved objects can be replaced, so the Object Pool Manager first checks whether any objects are free. Assuming there are unreserved objects, the Object Pool Manager next calls the StealObject( ) method to determine which object to replace. The Object Pool Manager calls the ReleaseObject( ) method to destroy the object to be

replaced. As a result, the pool now contains one slot to fill, and the CreateObject( ) method is immediately called.

Queuing a request: The fourth phase of processing occurs under the following conditions:

If no matches are found, if the pool is full, and if none of the objects is free to be replaced. Under these conditions, the Object Pool Manager queues the request.

The request waits until an object is returned to the pool. Pooled objects are returned to their originating pool when a client is done using them. If the waiting period is within a maximum allowable idle time, then the request processing starts over from the beginning, with an attempt to match the newly returned object. The maximum allowable idle time is configurable through the MaxWait variable in the registry.

Returning a FAILURE state: If the request is queued for longer than the MaxWait time, the request finally fails. Frequent request failures may indicate a need for the server administrator to review an object pool's configuration settings.

Maintenance features of the Pool Manager: In addition to processing object requests, the Pool Manager is also responsible for maintenance of the Object Pool. The Pool Manager prevents session time-outs and destroys physical objects in the pool, based on idle time or usage limits.

What the Pool Manager can do : In the context of Portlet development we can use the Pool Manager to manage, for example, sessions and associated objects. The following scenarios illustrate the advantages of using the Pool Manager:

One user, multiple Portlets

In this case pooling session objects will help to reduce the access time to a page since the session object is created only once and then shared by all Portlets.

User group, multiple Portlets

This shows the efficiency of the Pool Manager best. Besides sharing sessions for users of the same group, we can share view objects as well.

On Portal places we will find that this is a common scenario. Users having access to a Place usually have the same access rights to the underlying business applications as well.

One user, one Portlet

In this case we do not have any advantage, aside from the fact that implementing the Object Pool once is going to reduce coding time for Portlets in the future.

Performance pitfalls: The following recommendations are given to help avoid some common performance pitfalls that are not always easy to recognize.

Do not spawn threads

Since Portlets are multi-threaded to begin with, spawning child threads can create problems with synchronization or multi-threaded access to the spawned threads. Threads should be used with extreme caution, and when necessary, should be used briefly (no long-running threads, especially any that outlast a request).

Do not use threads to access J2EE resources

In certain Java Virtual Machine implementations, such as on zOS, there are a limited number of threads in the process thread pool which are allocated for accessing J2EE resources, such as JCA connectors. All such resources should be manipulated on the calling thread to minimize the possibility of starving the thread pool.

Limit temporary storage

Temporary storage includes temporary variables created within a block of code, which are used and then discarded. Temporary variables are typically utilitarian in nature, such as Strings, Integers, Booleans, Vectors, and such. However simple in nature, temporary variables take CPU cycles to create and destroy, and they occupy space on the heap which must be maintained by the garbage collector. The following is a list of simple guidelines for reducing or optimizing temporary storage:

- Reuse temporary variables as much as possible.
- Declare temporary variables outside loops.
- Instead of String, use StringBuffers, which are optimized for parsing and string assembly.
- Declare collection classes (Vectors, Arrays) with an initial size that is the average maximum size, to prevent growth and reallocation of space on the heap.

Avoid synchronized methods

Synchronization causes a bottleneck through our portlet, or a path that only allows one thread at a time to pass through. Besides slowing the entire portal system, the possibility of a deadlock occurring is also high, which could hang the portlet, or portal, for all users.

Avoid long-running loops

Simply put, Portlets need to be fast. Long-running loops consume a large number of CPU cycles and cause the portal page to wait for this one portlet to finish. If a long-running loop seems necessary, re-inspect the design to see if it can be accomplished by some other means, such as through block/notification logic, or breaking the large loop up into several shorter ones.

Use JSPs instead of XML/XSLT

JSP are more efficient than XML/XSLT in general. Since JSPs are compiled into servlets, they run much faster than having to parse XML and apply XSL style sheets on every request. Also, the portal infrastructure is optimized around JSPs, allowing for easy expansion into other markups, languages, and even browser support by simply adding new directory structures of JSPs. In general, XML parsing and XSLT processing is expensive. XSL processing, for example, causes hundreds of transient String objects to be created and destroyed, which is expensive in CPU cycles and heap management. If XML/XSLT is a requirement, then make use of caching as much as possible to reduce the amount of parsing and XSLT processing which must take place.

Chapter 6: Portalization Approach for Existing Applications

Application Integration

For most companies it is essential to be able to seamlessly integrate existing business applications and processes into their Portal. WebSphere Portal provides many options for business application integration, and no one method fits all needs. Using a portalizing strategy can assist in making the right choice for specific requirements.

The portalization process

Let's imagine typical business applications, for example, HR or collaborative applications. They need to be "portalized" into Portlets, which can then be made available for different Portal pages. An application can be represented by a single portlet, but there is usually not a one-to-one relationship between applications and Portlets. Most cases exhibit a one-to-many relationship. Portal applications are usually made up of multiple Portlets that use portlet cooperation. This way, application functionality exposed in Portlets can be combined in many different ways, sometimes allowing the user to put information in a context that not even the developer of the individual portlet has thought about. It is also important to understand that the same portlet can be used on many portal page contexts with different user roles.

The portalizing challenge

Thinking about the portalizing process raises the following questions:

- What portlet development techniques are available?
- What is the effort required for portlet development?
- What functionalities can be implemented within a portal?

- How can scalability and performance be ensured?

In the following discussion we address these questions and describe the factors and dependencies of the portalization process. As with any development project, there are multiple factors and dependencies that influence a portalizing project. The important parameters for a portlet project are:

- Business application: the characteristics of the business application.
- Integration techniques: knowledge of all the different options.
- Functional requirements: define the portal integration depth and user interface requirements for the portlet.
- Considerations: overall project parameters like resources, skills, and so forth. We need to develop a good understanding of the different factors and their dependencies to be able to successfully portalize a business application into Portlets. Therefore, in the following sections we drill down deeper into the various factors and dependencies.

Business applications

The most important piece in the portalizing process is the application itself.

We need to consider the following parameters to characterize the application:

- Size of the databases/number of records
- Number of users, maximum number of users
- Usage pattern, number of concurrent users
- Application complexity
- Integration of single or multiple backend systems
- Already Web-enabled or rich-client-based?

- Notes client-based?

Figure 7 can help to characterize a given business application.

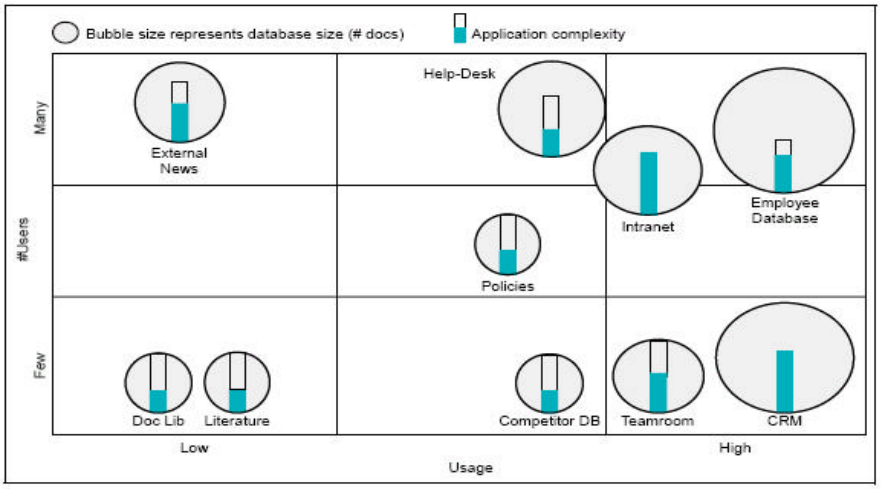

Figure 7 Business application types

It shows several different business applications, with their database usage on the horizontal axis and the number of users on the vertical axis. The size of each bubble represents the database size, and the bar in the middle indicates the application complexity. Placing a given application into this grid can help to develop a better understanding of the usage scenario for a portlet. This then allows us to estimate the workload on the portal and backend servers. Another important consideration is that the portal introduces new usage scenarios for our applications. With a rich-client application, a user usually accesses applications sequentially, meaning one at a time. On a portal page there are multiple portlets active, which will concurrently access our applications. This increases the server load significantly. In addition, the portal will broaden the user community for our applications because they are more visible now. All this can have significant implications for our backend systems and performance.

Performance considerations

Portlet performance is the most important factor in a portlet development project.

Let's imagine the following scenario. We are going to build an employee directory that has the following requirements:

- Employee directory with 100,000 entries
- Initially the portlet shows the first 10 Employees of the A-name category.
- Users can page through the Employees like in a paper phone book, or search for an Employee by Name or Employee Number.
- Portal user community of 40,000 users

We assume a concurrency of 5% of the total portal user number for the portal page the portlet is placed on. That means we have approximately 2,000 concurrent users for the portlet.

Let's make some further assumptions about our scenario, as follows:

The portlet is available on the portal home page for all users. Monday morning at 8 AM all employees are going to log on to the portal.

The interesting question now is: What is going to happen to the overall portal performance and server load?

Let's analyze: We are going to have 2,000 concurrent requests to display the initial view of the portlet. Creating 2,000 concurrent sessions to the backend system and rendering the first page for each user does not seem feasible.

If we planned for this scenario, we probably would have thought about caching strategies and overall session management for this use case. If not, this portlet

can significantly decrease the overall portal performance and put a very high load on the backend systems.

Note: Portlet performance and scalability is often underestimated in portlet development projects. It is good practice to do performance testing right from the start of the project. Performance is not only dependant on our coding or the backend systems, but also on the overall portal infrastructure.

Integration types

Portlets can be as simple as data display windows into existing applications, or as advanced as a replacement for complex workflow applications. They also can have different levels of integration with the portal. Integration types will help us to classify what integration depth we are looking for; this information is relevant when deciding what integration technique to use. We describe the following types including:

- Link
- Display
- Integrated
- Migrated

As for any development project, it is best practice to establish use cases for the portlet usage. We need to ask:

- What is the use case from a user's perspective?
- Does the use case vary with the role of the user?

Once we have this information we can decide what integration type our application falls into.

Link: The Link type is the easiest of all integration types. It provides the user with a link that launches into the native application. This can be either a rich-client-based or a browser-based application. Depending on the setup of the authentication system, the user may need to log on to the specific application. The rich-client will challenge the user for their password if the portlet is not configured to use the user's operating system login. If the application is browser-based and single sign-on (SSO) is enabled, the re-authentication is not necessary. Link-type integration can be achieved by using, for example, the Bookmark Portlet or Quicklinks Portlet that ship with WebSphere Portal.

Display: In the display category are Portlets that only display information from backend systems. If the user needs to interact with the application functionality, they must launch either into the rich-client-application or a browser interface. This can be a good option if an application is already Web-enabled. In the portal we can give the user an overview of, for instance, all the expense reports that he needs to approve, and then for the approval process launch the expense report application in a new browser window. Depending on the implementation, it would be possible to launch immediately into the expense report that the user wants to approve. This example illustrates that it is not necessary to portalize the whole application to the portal. Sometimes a small view into the application might be enough.

Integrated: An integrated scenario lets the user perform tasks in other applications inside the portlet, rather than having to launch the other application. In the previous scenario we could provide the user with a list of expense reports

to approve, but to perform the approval task we had to launch into a different application. Launching into a separate application might not be a problem, but being able to perform the task inside of the portlet context enhances the user experience and the usability of the portlet.

In our example, we want to allow the user to approve or deny an expense report.

In the denial case he should also be able to provide a reason. For a different user role, we also want to allow the creation of new expense reports from within the portal. In these two use cases we have exposed only some of the functionality of the expense report application to the portal. Other workflow processing of the expenses and backend integration with an HR system will remain untouched in the native business application.

This example shows the advantages of the coexistence of Portlets and existing business applications. Only the information and logic needed for certain use cases will be transferred into the portlet.

Migrated: A migrated portlet is one that is used to replace an application and transform entire business processes into Portlets. Usually an approach like this would involve a redesign of the application.

Considerations

There are a number of factors that have great influence on a project. Some of them are:

- Time frame for the project and milestones
- Available people and resources

- Available or set products and versions
- Target platform, such as AIX, Win 2000, and so forth
- Scalability and performance requirements
- Experience and development skills
- Portal infrastructure

Most conditions for a business application integration project are not any different than for other software development projects. Therefore, we should apply the same project management and development methodology to a portalization project that we would use for other development projects.

Chapter 7: Summary and Conclusion

Summary

Today's rapid growth of the intranet content has given users too many information resources as well as too much unstructured data. Users need new tools to better organize the available information found in corporate and worldwide resources, and better user experience. This thesis presented a web portal architecture and view from a developer's perspective and explained by practical examples using IBM WebSphere V5:

In chapter 1 we discussed about the web portal, web portal solution and about the web portal strategy. In Chapter 2 we explained the web portal advantages, web portal types, web portal software functions, web portal universal features and web portal user interface. In Chapter 3 we threw light on the importance of web portal strategy, about web portal concepts and technologies, the IT environment, building a business case, information technology analysis and results.

In chapter 4 we have shown that business application integration into the portal is useful for building Portals. Existing applications can be reused and exposed inside of the portal, allowing the display of existing information in new contexts.

Bringing together the best of the Portal and existing business applications allows us to build a new class of hybrid applications. We outlined a transformation strategy to achieve successful Application-to-Portal transformation. It helped us to understand the integration challenge and to carefully choose the best integration technique and development approach for a given business application.

It addressed the design stage of the solution development process, highlighting the key issues that are specific to portal and application designs.

It is important to note that moving an existing web application to a portal doesn't have to take place all at once. It can be a multi-step process. The first step is to present the existing content or functionality from within the portal. Once this is accomplished, the application will need to be better integrated with the portal such that it can leverage the portal navigation, personalization, and security mechanisms. We should also evaluate existing data and knowledge sharing issues before hand, those include;

Do employees share common corporate semantics throughout the organization?

Do employees have access to the data sources and knowledge experts needed to complete their jobs?

When different groups of employees access data in the organization, do they all see consistent data values?

Can employees easily find the information and data sources they are looking for?

Are employees aware of what information or data sources are available for them to access?

Can intranet and extranet data sources be searched to locate information or data sources employees are interested in?

Can employees with similar job responsibilities share experiences and ask questions to each other in the organization?

Are employees made aware of information, knowledge experts, and data sources in other departments or other employee roles that directly affect their specific responsibilities?

Can employees easily navigate through information available in the organization?

Can employees isolate issues and coordinate a resolution in a collaborative environment with other decision makers?

Do employees with similar job responsibilities make consistent decisions?

Can employees personalize their work environment?

Does the organization foster a shared community?

Do employees have easy access to summarized information and reports?

Can employees drill into information available in these reports?

Are employees made aware of information that would be helpful or relevant to their assigned responsibilities?

Is information that is available to employees current and relevant to the decisions they need to make?

Finally, the ultimate step in portalizing the application is to make it a web service such that the actual application functionality doesn't need to be co-located with the portal framework itself.

Conclusion

This thesis has discussed a number of strategies for enabling an existing application to be adapted to a web portal (portalized). In doing so, the goal has been to understand the advantages by providing the portal architecture and necessary concepts for a developer to plan for and ultimately perform the portalization. Hence showing that how portals are better than scattered applications. I conclude with some important observations found during the study.

• People issues are the biggest cost in portal implementation

• Unhappy users mean low ROI

• Involving users from an early stage eases acceptance problems

• Results of usability testing can help convince the skeptics

• A cross-departmental steering group makes users feel they have a voice, and helps overcome departmental resistance to the portal

• An independent speaker can help sell the benefits of a portal

• Withdrawing funding for non-portal-compliant intranet projects helps speed up standardization

• Different CMS and portal products have different strengths and weaknesses; make sure to decide what priorities are before making a choice.

• How will the underlying technology platforms in the portal affect consistency of the user interface?

• Think about the different ways content might be used once it's out in the Portal.

• Departmentally-generated content may need central guidelines to make it usable. Rules can make users lives easier in the long run.

• Usability costs extra upfront but pays dividends in the long run.

• Results of usability testing can be used to win over dissenters as well as improve portal usability.

• General user surveys worked well for setting direction and highlighting problem areas

• Not all portals have a single home page. In some cases, they may be unified by common navigation, not a common homepage

• Good portal design doesn't have to mean fancy design. Busy users may prefer to get their jobs done quickly and go home, not play games and personalize their pages.

• For most corporate users, content takes priority over style

• Portal design will however reflect corporate culture

• International organizations will need to build portal information taxonomy that works globally, not just locally

• Introducing information standards is more challenging than imposing design and layout standards

• It may help to have a specialist editor in each department to submit information to the portal

• Many users don't bother to personalize pages for themselves, and are happy to accept default settings

• It can be hard to decide which information users need to see. By using personalization to de-emphasize rather than hide less relevant information, leave people's options open

• Role-based personalization will often suffice to give users the options they truly need and will free them from having to fiddle with customization settings

• Before letting IT go ahead and choose a single sign-on product, check how well it will work with most important legacy applications

• If portal incorporates third-party applications or services, this will probably complicate the single sign-on process

• If there are a lot of legacy applications to bring into a single sign-on infrastructure, a practical approach is to start with the ones that are used most frequently.

• If planning is to use a portal product, remember to check out how its search feature works

• Generic search isn't appropriate for every lookup task. Consider adding specific searches for specialized tasks.

• Users generally prefer free-text search

• But keyword search can be better for task-specific searches as long as content providers are careful about entering the right keywords.

• It's important to decide on measurable goals before starting the portal project

• By facilitating repurposing of content, portals can sometimes be used to generate positive revenue.

==== The End ====