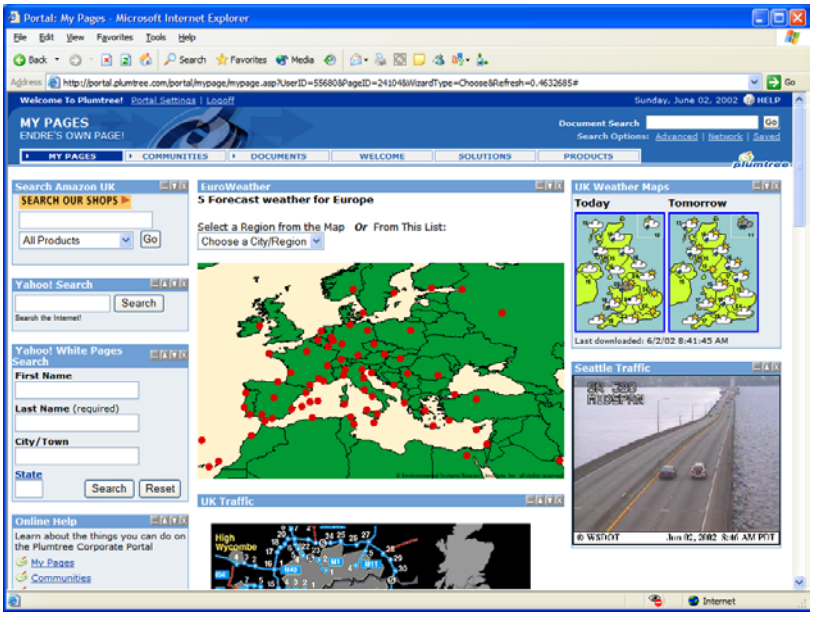
Modular Development Frameworks for Corporate Portals

- A Literature Review

Endre Stølsvik, June 2002.



A screenshot from the Plumtree Corporate Portal

Acknowledgements

I would like to thank my mentor M. Letizia Jaccheri for guidance and comments throughout the semester, my boss and mentor Eirik Rasmussen for reassurance during tough times, Siw Gabrielsen for helping me out with my English, Raphaël Luta and Glenn Golden from the Apache Jetspeed team for answering my questions, my flat mate Per Krämer for support, and finally Ine Vasshus for keeping my spirits up!

Sandnes, June 5, 2002

Endre Stølsvik

1 Summary

This thesis investigates the functionality of current development frameworks for *corporate portals*, a relatively new concept also known as *enterprise information portals*. The main focus of the study is on areas concerning portals' functional components, their architecture and architectural features, and frameworks' concepts and features.

The thesis is based on a review of recent, topic-related articles and scientific literature, in addition to analyses of six vendors' corporate portal framework. The analyses are based mainly on the vendors' white papers, framework documentation, and technical documents. However, two of the portal systems have furthermore been subject to installation and testing, and the test results serve as additional basis for analysis.

In essence, the thesis is divided into four parts. Part one constitutes a review of what corporate portals are, what arguments are being used for their value, what requirements they should fulfill, and which functional components they usually provide. This part is mainly based on the literature review. The analyses of the portal framework solutions are presented in part two. In part three, the findings of the analyses are summarized and discussed. Finally, part four attempts at predicting how the development of corporate frameworks will evolve over the next few years.

Given the recent initiation of standardization efforts aimed at corporate portals, and taking into consideration the significance of these efforts to the commissioners of the thesis, it was furthermore decided to investigate these efforts, and evaluate the implications they may potentially have for the corporate software industry.

The thesis concludes that a further dissemination of corporate portals, and particularly a standardization of corporate portal frameworks, may in the longer term potentially change the way in which small and large corporations employ information technology. Corporate portals may be viewed as a new type of corporation wide operating system. By installing a productivity or enterprise application on the corporate portal, all of the corporation's employees will – relative to their corporate role – have instant access to all of the application's features.

2 Table of contents

1 Summary 1
2 Table of contents 2
3 Introduction 5
3.1 Motivation 6
3.2 Scope 6
4 Definitions 8
5 Corporate Portals Explained 10
5.1 The Portal User Experience 11
5.2 Corporate Portal Arguments and Requirements 12
5.2.1 Make Information Accessible 13
5.2.2 Reduce Information Overload 14
5.2.3 Information Aggregation 15
5.2.4 Enter Data, Enterprise Application Integration 15
5.2.5 Knowledge Management 16
5.2.6 Collaboration 19
5.2.7 Remote Access 22
5.2.8 Integrate the Extended Enterprise 23
5.3 Functional Components of the Corporate Portal 26
5.3.1 Taxonomy and Directory 26
5.3.2 Publishing and Content, Document and File Management 26
5.3.3 Searching 27
5.3.4 PDA integration and synchronization 27
5.3.5 Workflow and Business Process Automation 28
5.3.6 Alerts and Notifications 28
5.3.7 Push Information and Subscriptions 29
5.3.8 Users, Groups, Permissions and Roles 29
5.3.9 Single Sign-On 30
5.4 Portal Categorizations 30
6 Corporate Portal Framework Vendor Analysis 33
6.1 The Need for a Portal Framework 33
6.2 Vendor Selection 33
6.3 Analysis 35
7 The Vendors 37
7.1 Apache Software Foundation – Jakarta Jetspeed 37
7.1.1 Architecture and Architectural Features 39
7.1.2 Framework concepts 42
7.1.3 Framework features 44
7.1.4 Software Quality Model References 47
7.2 BEA – WebLogic Portal 48
7.2.1 Architecture and Architectural Features 49
7.2.2 Framework concepts 54
7.2.3 Framework features 58
7.2.4 Software Quality Model References 59
7.3 Epicentric – Epicentric Foundation Server 60
7.3.1 Architecture and Architectural Features 60
7.3.2 Framework concepts and features 63
7.3.3 Software Quality Model References 66
7.4 IBM – WebSphere Portal 66
7.4.1 Historical Review of the Portlet API 67
7.4.2 Architecture and Architectural Features 68
7.4.3 Framework concepts 70

7.4.4 Framework features ... 70
7.4.5 Software Quality Model References ... 74
7.5 Oracle – Oracle Portal ... 75
7.5.1 Architecture and Architectural Features ... 76
7.5.2 Framework concepts ... 79
7.5.3 Framework features ... 80
7.5.4 Software Quality Model References ... 82
7.6 Plumtree – Plumtree Corporate Portal ... 82
7.6.1 Architecture and Architectural Features ... 83
7.6.2 Framework concepts and features ... 86
7.6.3 Software Quality Model References ... 89
8 Architecture and Architectural Features ... 91
8.1 High-Level Architecture ... 91
8.1.1 Portal System as Enterprise Operating System ... 91
8.1.2 Architectural Standpoint: Webpage Assembly or Operating System ... 92
8.1.3 One or Two-tier Rendering ... 92
8.1.4 Parallelizable Portlet Rendering ... 94
8.1.5 Portlet Caching ... 95
8.2 Platform and Language ... 95
8.2.1 Dominant Language and Platform: Java and J2EE ... 95
8.2.2 Vendor Lock-In ... 96
8.2.3 Integration with J2EE ... 97
8.3 Rendering ... 98
8.3.1 Layouts ... 98
8.3.2 Portlet Modes and Portlet Rendering ... 99
8.4 Configuration and User Management ... 99
8.5 Preinstalled Modules ... 100
9 Framework Concepts ... 101
9.1 Customization, Preferences and Personalization ... 101
9.2 Security ... 102
9.2.1 Access Control ... 102
9.2.2 Single Sign-on ... 102
9.3 Administration and Delegation ... 103
9.4 Modules and Portlet Applications ... 104
9.5 Services ... 104
9.6 Multiple Workbenches ... 104
9.7 Invitations ... 105
9.8 Remote Portlets ... 105
9.9 Internationalization ... 105
9.10 Statistics and Business Metrics ... 105
10 Framework Features ... 106
10.1 User Abstraction Object ... 106
10.2 Multiple Markup Support and Client Capabilities ... 106
10.2.1 Multiple Markup Languages Support ... 106
10.2.2 Client Capabilities ... 107
10.3 Portal Creation ... 107
10.3.1 Multiple Portals within a Single Portal Server ... 107
10.3.2 Portal Creation ... 108
10.4 Portlet Creation ... 109
10.4.1 Developing Portlets using APIs ... 109
10.4.2 Editor Based Portlet Creation ... 110
10.5 User / Portlet Communication ... 111
10.5.1 Portlet Output to User ... 111
10.5.2 Input from User to Portlet ... 112
10.6 Messages, “Inter Process Communication” ... 113

10.7 State, Persistence, Preferences and Configuration 113
10.8 Roles and Permissions 114
10.9 Internationalization 115
10.10 Caching 115
11 Software Quality Model 116
11.1 Functionality 116
11.2 Reliability 117
11.3 Maintainability 118
11.4 Portability 119
11.5 Scalability 121
12 Portal Framework Evolution 122
12.1 Standardization Efforts 122
12.1.1 Consequences for not having a standard 122
12.1.2 Java based Portlet API 122
12.1.3 Web Services Portlets 123
12.1.4 Possible features included in the standardizations 126
12.1.5 Consequences 128
12.1.6 Potential Difficulties 129
12.1.7 Vendor Diversification 130
12.2 Enterprise Operating System 131
12.3 Other Technologies 132
12.3.1 Microsoft .NET 132
12.3.2 Other Languages 133
12.3.3 Non-browser Based Portals 133
13 Conclusions 135
14 Appendix A: Definitions used from ISO/IEC 9261-1 136
14.1 Definitions 137
14.1.1 Functionality 137
14.1.2 Reliability 137
14.1.3 Maintainability 137
14.1.4 Portability 138
14.2 Comments 138
15 Appendix B: Java 2 Enterprise Edition 140
16 Appendix C: References and Resources 144
16.1 Papers and Articles 144
16.2 Other Resources 145
16.3 Vendor specific Documents and White Papers 147
16.3.1 Apache Jetspeed 147
16.3.2 BEA 147
16.3.3 Epicentric 148
16.3.4 IBM 148
16.3.5 Oracle 148
16.3.6 Plumtree 149

3 Introduction

The terms *Corporate Portal* and *Enterprise Information Portal* refer to a relatively new concept in the information technology business, primarily relating to the management of corporate information and knowledge. It is a genre of web-based applications, meant for users both internal and external to the corporation. IT consultants and corporate portal vendors promise easy, configurable access to all of the corporation's own internal information and knowledge, along with information from external resources, all in a structured, easily comprehensible manner.

The popular *web portals*, whose concepts were created by Yahoo! with their MyYahoo Service (http://my.yahoo.com/), have heavily inspired the user interface for most corporate portals. Requiring only a web browser as the front end to the system, the idea is that the users can select which of multiple resources they would like to have on their web-based desktops, much like the windowed user interface of most modern operating systems. The users can further configure these often-called *portlets* to suit their information needs from each underlying information source.

Arguments similar to the ones given in favor for corporate portals were earlier used for *corporate intranets*, the company internal web sites used to gather, share and distribute information and knowledge. However, today many corporations are faced with a multitude of corporate intranets. Each of them supports different aspects of the corporation's information needs, e.g. a document intranet, a directory of often-used files and forms, and several intranets that web-enable information to and from the underlying systems. This jungle of information sources might just create yet another informational and navigational overload. The corporate portal, on the other hand, offers a one-stop, single-point-of-access resource, where all corporate information is aggregated, filtered through a *role* and *access control* system, and finally personalized according to the user's preferences and job situation.

Lately, most of the computer software vendors have delivered their version and vision of a corporate portal; IntranetFocus [IntranetFocus, 2002] have identified over 70 vendors. This list includes the software giants like IBM, Oracle, SAP and Microsoft, but also a number of companies that were born with this market. Several of these vendors have very specialized portals, with a preconfigured set of portlets, often interfacing to only one or a specific set of back end systems. Others are much more configurable, or preferably, programmable, providing means for extending the system with new portlets or extending the integration by developing new connectors towards underlying systems.

This latter type of portal systems provides for a modular type of framework for portal development. It is modular in the sense that each portlet is a separate, independent part of the system, enabling corporations to assemble and develop the functionality needed in a piecewise and gradual manner.

These extendable systems provide a host of different features for extending and configuring the product for the administrator or programmer. They are all very different, though; even the basic architecture of the systems varies considerably from product to product.

3.1 Motivation

CoreTrek is a relatively new company with headquarters in Sandnes, Norway. It was founded in 1999 by a group of investors from the Hanabryggene Technology Centre.

One of their earlier projects was of such a scale that the project manager decided to make a foundation on which to build the rest of the project. The aforementioned website MyYahoo and Netscape's Netcenter (http://my.netscape.com/) inspired this foundation. Initially meant as a small platform for this one project, it was quickly realized that the direction in which this project was heading could give benefits for other projects as well. CoreTrek divided the project into two separate projects, one that focused on the application, and one that focused on the foundation itself. This would enable the programmers to develop the foundation further, tending to needs from multiple projects.

After some time, CoreTrek realized that other companies had come up with the same idea. Not only was the technology with which both MyYahoo and Netcenter was built upon available for hire[1], but several new actors was emerging in the field. This segment of the computer industry had even been defined and terms had been coined.

This foundation has been further developed, and the *Machete* server, as it was named, has now been released in version 1.1. The portlets that populate the W*orkbenches* in this framework are called *Corelets*. A collection of Corelets, which together form a functional unit, is called a *module*.

The framework's focus has been application development, and is not restricted to any specific underlying system. It is Java based, running on any Java based web application server, and may interface with any database. A Corelet API provides developers with the means to program new portlets and modules easily. Several standard modules have been developed to enable a fast assembly of a new portal deployment.

Due to the lack of any existing official or de-facto standards in the portal and portlet area, CoreTrek have developed their portal and portlet API independently, as have most other portal vendors. CoreTrek have so far not been able to fully analyze the emerging technologies and trends in this field, nor been able to do an analysis of where efforts have been focused by other vendors. The hope is that this thesis will shed some light on these topics.

3.2 Scope

This thesis' objective is to perform "a review of available recent literature about modular development frameworks for corporate portals, with the purpose of charting current and future trends in functionality provided by these types of frameworks".

[1] MyYahoo is built upon technology from the company TIBCO. Yahoo! and TIBCO have co-developed a portal system they call Yahoo! Portal Solutions, using the Yahoo! PortalBuilder (http://enterprise.yahoo.com/portal/products/). The Netscape Netcenter solution was available as a hosted service, the Custom Netcenter Hosted Service, and later developed as a stand-alone server in partnership with Sun. This partnership used the iPlanet brand, and their portal server was called iPlanet Portal. Netscape was later acquired by America Online, AOL. Sun have recently (March 17, 2002) concluded its alliance agreement with AOL, and their portal is now called Sun ONE Portal Server (http://wwws.sun.com/software/products/portal_srvr/home_portal.html).

The intent with this commission was to give CoreTrek ideas and guidelines of where to focus further development of the Machete framework, observing that no actual standard existed. This would be done in an effort of trying to align the Machete portal server with such trends. Somewhat unexpectedly, several standardization efforts were initiated during the writing of this thesis. As standardization is a very important juncture for a software vendor in any software market, some effort has been directed towards determining what these standards implicate. This has however left the author with a somewhat shifting scope.

The available scientific literature with corporate portals as the main subject is rather scarce. However, a considerable selection of articles from business and computer professionals' magazines exists, which combined with the scientific literature ensures that the background information is adequate. The literature found revolves around the more high-level aspects of a corporate portal; what are corporate portals, what requirements should they fulfill, and what functional components do they usually offer. Chapter 5 – "Corporate Portals Explained" is mostly based on a review of this literature.

The thesis further charts what functionality a selection of development frameworks for corporate portals currently provide. White papers, data sheets, programming information and other documentation from each of the selected vendors are studied and analyzed in Chapter 6 and 7. Two of the chosen frameworks was also installed. The findings in each analyzed category is summarized and discussed in Chapters 8 through 11. Finally, Chapter 12 will give some thoughts on how corporate portal frameworks will evolve.

Functionality is a wide term. The reader should recognize that this thesis' focus specifically is the development framework, and not the portal system as a whole. Functionality is thus defined to include the *functional components, architecture* and *architectural features,* and *framework concepts and features* of a corporate portal development framework. These terms are further defined in the Chapter 4 - "Definitions"[2].

[2] The international standard (*ISO/IEC 9126-1) Information Technology – Software Quality Characteristics and Metrics* defines the software quality characteristic *functionality* as "The capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions", whose sub-characteristics are *suitability*, *accuracy*, *interoperability*, *compliance* and *security*. It is important to remember that this thesis' scope is to assess the functionality of the frameworks, not the actual corporate portals. This thesis' definition of functionality diverges somewhat from this standard. See also appendix A, "Definitions used from ISO/IEC 9126-1".

4 Definitions

Definition and explanation of some important and frequently used terms:

- *Architecture* refers to the architectural building blocks that constitute the portal system. This includes the hardware, the operating systems and any application servers and databases necessary to run the system. It also refers to how the portal system itself is constructed; for example, how the portal server runs and handles the portlets, and how the user actions are communicated to the portlets.

- *Architectural features* refer to the features of the vendor's selected architecture, often features that are intrinsic of such architecture, or that are rendered possible using such architecture. For example, whether the rendering of all the portlets comprising the workbench is done sequentially or in parallel, and whether user groups are supported.

- *Concepts* or *framework concepts* refer to high-level concepts of the framework. For example, "user roles", "theme based presentation" and "multiple customizable workbenches". This term does not include application specific or application related features like "group calendar" or "discussion board".

- *Framework features* refer to the features of a development framework; especially lower-level programmatic and declarative design features and capabilities. For example, programmatic features like "Java based portlet API" and "scheduled task dispatching", design features like "WYSIWYG menu creation", or declarative features like "portlet parameters declared using XML".

- *Framework* and *portal framework* means a development framework in which one builds a Portal. This includes frameworks where one declaratively configures a portal and frameworks that are geared towards programmers where one can, using APIs, program new portlets or extend other features of the portal system. A portal systems that merely supply a predefined, non-extendable set of portlets from which the user might choose a setup, are considered not to have any framework.

- *Functional components* refer to the application-like functions of a portal system that often would be present in a lower layer of the architecture of the system. The functional components will often be used by several application developed on the portal, like some embedded service. For example, an application could be a group calendar, while a functional component could be the user notification system, used by the calendar application when a deadline is approaching.

- *Java 2 Enterprise Edition, J2EE,* is a Java application server standard developed by Sun Microsystems Inc. in collaboration with several leading software vendors. Appendix B contains a review of this standard. Several of the frameworks analyzed builds upon this standard.

- *Modular development framework* refers to a development framework that enables the programmer or designer to develop applications and functional components in a modular way. Each module may be designed and programmed individually. The full system is assembled using these newly built modules along with standard modules that come with the framework.

- *Module*, also called *Portlet Application,* refers to a set of portlets, which often forms a functional unit. A module might be anything from full applications like calendaring and sales automation systems consisting of multiple portlets, to small utility modules, for example, a single portlet news-headlines fetcher.

- *Portal product*, *portal system* and *portal* refers to a *corporate* portal system, except when otherwise specified.

- *Portlet* refers to "a piece of a workbench". It often remebles a window from most modern operating systems, and contains for example close and minimize buttons to control the visibility of the portlet. Portlets are sometimes self contained, acting like one small program in itself, but are often used in conjunction with other portlets. The set of portlets then forms a module.

- *Workbench* refers to the users' view of the portal. Portlets populate the workbench, and it often resembles a *desktop* from most modern operating system. Most portal system divides the workbench into several columns based on how the rendering of a web page is done. This restricts the possible widths of a portlet to the widths of these columns

- *Physical Server* is used when a distinction between the server software process and the physical processing unit ("machine") is needed. The physical server refers to the processing unit, while server or logical server is used for the software process.

Some terms from the standard ISO/IEC 9126 – "*Information Technology – Software quality characteristics and metrics*" is used throughout this thesis, and is listed and explained in Appendix B.

5 Corporate Portals Explained

Corporate Portals, or Enterprise Information Portals, is a new concept in advanced intranet solutions. Shilakes and Tylman [1998], two industry analysts working for Merrill Lynch, coined the term "Enterprise Information Portal" in their business report dated 16th of November 1998, and thereby defined this new "investment space". They define such a system as:

> "Enterprise information portals are applications that enable companies to unlock internally and externally stored information, and provide users a single gateway to personalized information needed to make informed business decisions"

In the same report, the authors further consider enterprise information portals as:

> "…an amalgamation of software applications that consolidate, manage, analyze and distribute information across and outside of an enterprise (including business intelligence, content management, data warehouse and mart and data management applications)."

Several terms and definitions are used frequently[3]. Both vendors and analysts attempts to give their own view or nuance to the different terms in use, or coin new terms. Firestone [1999] have done a survey of the different terms and definitions, and states that "the process of definition is a "political" business – an attempt to persuade the Investment/IT and ultimately the user community to define EIP in a manner favoring one's own vendor or analytical interests." Vendors are eager to have their definition accepted as the prevailing one, thus branding the other vendors' products as inferior. Analysts would get ahead of over other analysts and consultants if their definition were the most cited one.

The terms corporate portal and enterprise information portal have become the most frequently used, and they are used interchangeably. Despite the different terms and definitions, every analyst and vendor recognizes the work of Shilakes and Tylman, making their definitions the most cited in the field. Still, there has been some critique over their lack of collaboration focus in their definition [Dias, 2001]. Several analysts would like a corporate portal definition to place a stronger emphasis on the cooperation between users and generation of information.

This chapter is further divided into four subchapters. The first explaining in detail how the portal, portlets and workbenches are experienced by a user. Secondly, a set of arguments for the existence of corporate portals is set forth, along with the requirements these arguments lead to. The third subchapters details a set of functional components typically found in corporate portals, while the last subchapter explains a type of categorization of the portal systems useful for the selection of vendors in the vendor analysis.

[3] "Enterprise Collaboration Portal", "Enterprise Knowledge Portal" and "Business Portal" are also used, along with the more common "Corporate Portal" and "Enterprise Portal".

5.1 The Portal User Experience

All portals share two concepts, namely *portlets* and *workbenches*. The terms are often different from product to product, but these terms will be used throughout this thesis. A portlet is a windowed box on the user's workbench. The portal concept draws ideas from the already familiar windowed user interfaces from modern operating systems, while letting the user work in the familiar web browser interface. A screenshot from a typical layout is shown in Figure 1.

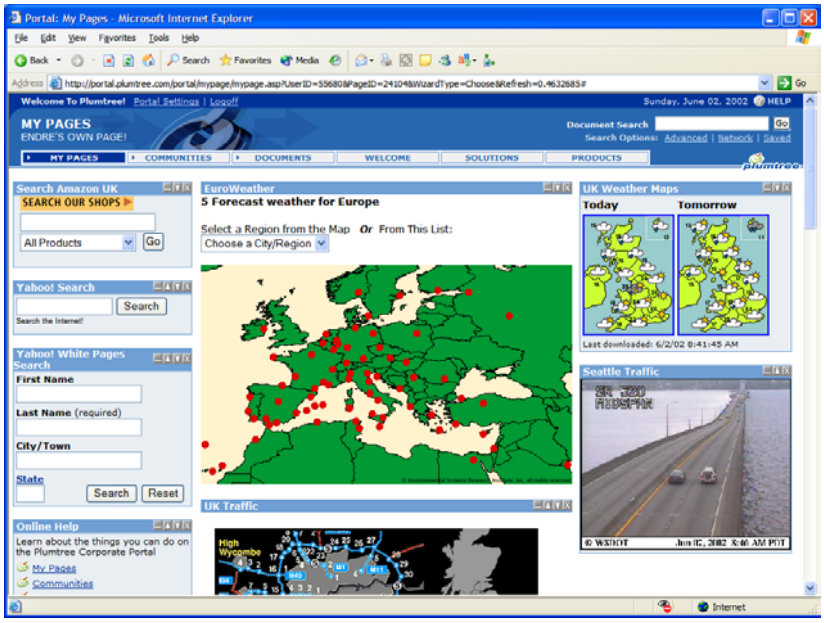


Figure 1 View from Plumtree Portal, demo version available online

The portlets are placed on a workbench by the means of some configuration scheme. Some portals use selection boxes where the user ca see a textual representation of the portlets that appear in each of the columns that exists on the workbench, and then a selection box from which to add portlets. The user can then change the layout of the portlets by moving the already existing portlets between the columns, or add or remove portlets to and from the columns. A user interface for this style is shown in Figure 2.

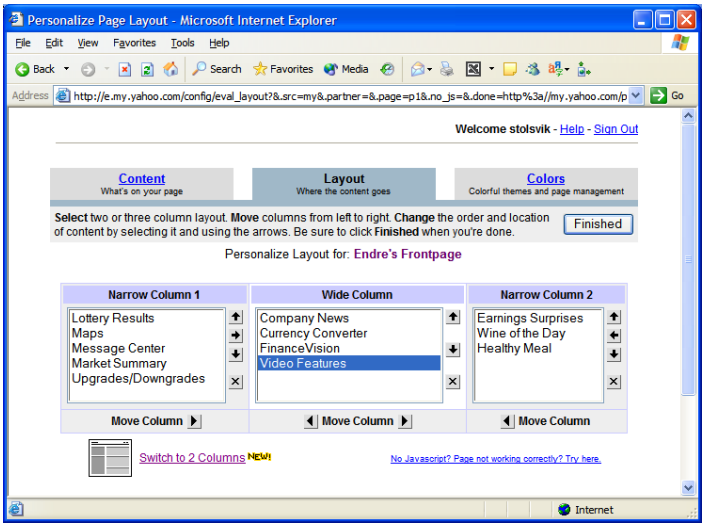


Figure 2 Selection-box based configuration. From MyYahoo, the original web portal

Another selection scheme shows the user a graphic representation of the workbench. The moving about, adding, and removing of portlets is thus a more visual-like experience. This scheme is shown in Figure 3.

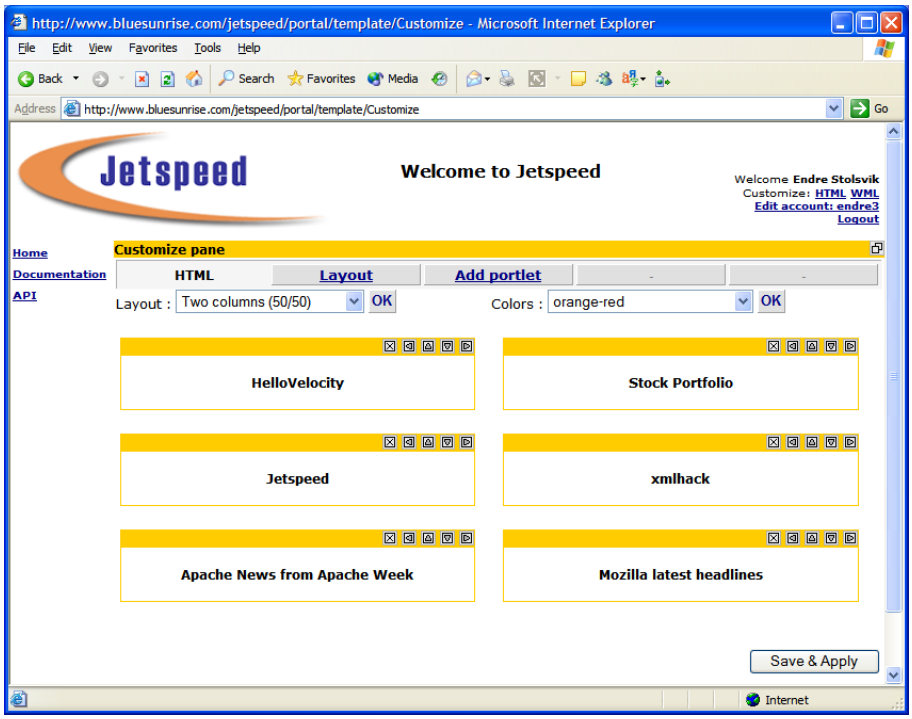


Figure 3 Graphical configuration of workbenches (Apache Jetspeed EIP)

Most portal systems enable the user to operate with several workbenches, switching between workbenches by clicking on their buttons or "tabs". The administrator of the system often have more control, being able to specify whether users should be able to configure their own workbenches, specify mandatory workbenches, and make configurations that locks down a set of portlets to a workbench, prohibiting the user to remove them.

This separation of the content inside of the portlets, and the frame which surrounds the portlet, including title bar, window control buttons and similar, makes the use of *themes* easy. A theme includes the design of the portlet frame, colors and background of the workbench, and the navigation bar. By using *Cascading Style Sheets* (CSS) and a strict convention of how to format the content of the portlets, one can also change the fonts, sizes and colors of the content with the theme. Thereby, one can change the entire *look and feel* of the portal by merely choosing another theme. This theme selection may be user selectable, with a default, and the administrator might be empowered to lock down a selection for the users.

5.2 Corporate Portal Arguments and Requirements

Every vendor and analyst has their set of arguments for how corporate portals can make a corporation run more efficient, to be more productive, and gain a competitive advantage. One ends up with a variety of arguments for the existence of these systems. A representative selection of these is presented in the following sections.

These arguments lead to several requirements that a portal system should fulfill. Several portal modules and functional elements that accomplish these requirements

are discussed alongside the arguments following, while others, of a lower-level nature, are presented later in Chapter 6.3 – "Functional Components of the Corporate Portal".

5.2.1 Make Information Accessible

The report from Shilakes and Tylman [1998] defines enterprise information portals as applications that enable companies to unlock internally and externally stored information. This is a major focus of all vendors and analysts. As a corporation grows older, it acquires enormous amounts of data, scattered in multiple data sources. These systems hold information about customers, orders, internal resources and much more. If a company merges with or acquires another company, even more data in additional sources are gained.

Many companies make frequent use of externally stored information. This might be online databases where industry standard information is stored, online newspapers and many others. As the internet still is maturing, more and more information will be available online. Table 1 contains some examples of internal and external information sources.

Internal
Contact information, e.g. customers, suppliers and partners
Sales organization; leads, prospects, contracts and agreements, drafts, business intelligence
Invoices, both incoming and outgoing
Stock / inventory
Orders
Documents, e.g. internal guidelines, product datasheets, white papers
Physical resources, e.g. cars, monitors, machinery
Human resources, knowledge management, expert help
People-to-people communications; e-mail, "chat", news / bulletin board systems
External
Online newspapers, magazines, weather
Company finances information, e.g. http://bizweb.no for financial information of all Norwegian companies
Online databases, e.g. shipping, chemicals and medicine
Journals (Science, technology and medicine, the so-called STM journals) and professionals' magazines
Online shopping, office supplies, travel and banking
Yellow and white pages, dictionaries
Map and driving direction applications
Government information, e.g. laws and regulations
Homepages to competitors, potential suppliers and customers

Table 1 Examples of internal and external information sources

Many information sources are difficult to access, known to or handled by only a few persons in the organization. For a mundane example, most companies need to focus on the customers who actually give them profitable business. It would thereby be beneficial if all employees in the organization knew, at all times, who the 20 most profitable customers are. This is most often not the case, because usually only

fractions of the company have the necessary access to and knowledge of the accounting system where such information is stored.

Dias [2001] states the problem in the following way:

> "Most of the time, this information is stored in computer hardware in a unorganized way, spread in databases, rendering access to relevant knowledge difficult, and compromising employees' productivity on their daily activities. Consequently, many modern enterprises lack a global view of their own data and information."

The problem results from the fact that the employees need to have a prior knowledge of both the existence and usage of the different resources, both internal and external. This might not always be the case; either because the resource might simply not be known to the employee, or the employee does not know where to look for it [Schroeder, 1999]. In addition, the user often needs some access privilege for the resource, usually some username and password that have to be remembered. All this makes for considerable obstacles; the employee must first aware of the resource, then request and gain access, and finally remember yet another username and password.

These difficulties may eventually lead to some relevant information not being considered for the case an employee is working on, in effect making the employee take an uninformed decision; suboptimal or even wrong.

A portal system can make both internal and external information accessible by providing consistent, intuitive user interfaces for each source, and by aggregating multiple sources into single views [Schroeder, 1999]. Searches can be done through multiple sources; thereby avoiding that the user misses important information due to lack of knowledge of some source. Making information resources readily accessible from within the portal itself will promote the usage of such information. Allen [1977 cited Detlor 2000 p.94] states that, "the convenience given by such accessibility can promote the acquisition and use of information throughout the entire enterprise as individuals tend to use information characterized by high accessibility".

5.2.2 Reduce Information Overload

Although making information accessible is an enormous advantage of the portal system, there is a problem at the opposite end of the spectrum. This is the *information overload* problem, where users are swamped with information from every corner in the enterprise [Grammer, 2000].

Thus, making information easy accessible is not the solution in itself. The amount of information that a user is required to run through in order to gain the necessary knowledge should instead be reduced. In addition, the amount of time required to find the exact piece of information needed should be minimized. Users must feel confident that they have found all the relevant information available.

In order achieve these objectives, key elements in a portal should include information aggregation, good categorization and navigation tools throughout the information sources, and proper searching tools that can search multiple information sources in one search.

5.2.3 Information Aggregation

A lot of information refer to other pieces of information, or make more sense when grouped and compared with other information. BEA states, "Customer information is one example of the type of information that often resides in multiple silos [of information] within the enterprise" [BEA E]. There might be a company record and several invoice records in the accounting system, a customer record and multiple agreement records in the customer relationship management system, several records in the support line system and probably even more records in other customer related systems. By having all this information readily available when a customer contacts the company, the employee will be able to provide customers with better service.

Business Intelligence and Decision Support

Business intelligence is a term used by the *data warehouse*, *on-line analytical processing* (OLAP) and *business reporting* industries[4]. The idea is that a sizable company usually sits on enormous amounts of information that never get used. These systems work by fetching data from multiple sources using so-called *connectors* and *adaptors* that can access diverse data sources, from main frames and legacy data to standard SQL based databases. This is often done by scheduled *batch processing*, for example every day or every hour. The data is then "cleaned" by defining single units and unambiguous concepts, for example, ensure that all sales are in the same currency, and "a purchase" is defined to be some specific event. Thereafter, the data is moved and arranged in a central data store. The data is said to be "time-variant", meaning that information about when an event occurred is stored.

The value of all these steps is that the data can at this point be aggregated along different *axes* or *dimensions*. For example, the "number of purchases" may be plotted against either "time", "place", "product group", "customer income" or some other dimension. This can uncover valuable information that previously was unknown to the organization.

The term *decision support* lies along the same lines. Business intelligence is used to facilitate decision-making by giving the user an overview of the totality of the data available in the corporation. Every aspect of the business decision processes, from long-term strategies to short-term decisions may benefit from having numbers, reports and statistics from the enterprise's different information sources readily accessible.

Schroeder [1999] points out that only a small fraction of users, so-called "power users" and the IT administrators themselves, have benefited from the many business intelligence projects that have executed. He argues that around 95 percent of the typical users in an organization either lacks access to these analytical tools, or lack the expertise to use them. Such reports and tools should therefore be made accessible in the portal, preferably at different user levels, from numbers and simple reports that are "one click accessible", to tools where the user needs to be an analytical expert to utilize them fully.

5.2.4 Enter Data, Enterprise Application Integration

Not only would one want to make information accessible by extracting data from one or multiple data sources and applications within the corporation. Being able to enter

[4] Resource site on data warehousing and related technologies: http://www.dwinfocenter.org/index.html

data into the different applications and systems, conveniently, integrated into the user's workspace, is just as valuable. This type of bidirectional integration is referred to as *Enterprise Application Integration*, EAI[5].

Making a whole enterprise application available from inside the portal can be a daunting task; however, this is usually not necessary. Some aspects of an application might be used much more frequently than other aspects, and large parts of the staff might never even make use of major parts of the application due to restrictions to their role in the company. Under such circumstances, it would benefit the organization considerably if only these frequently used parts were made accessible through the portal.

In other cases, parts of an application are used is used in a reduced manner, where several fields and even whole "tabs" of a data entry forms is left out in the daily activities for most users. In such cases, making a simplified of the application accessible through the portal makes good sense. In these reduced usage scenarios, some of the functionality could be altered and tailored to the specific needs of the organization. For example, some fields of the form could be left out, some could be pre-filled with the most used or default data, and others fields could have menus or other selection facilities attached. With carefully designed interfaces, simple and intuitive enough to be used by anyone, the corporation can avoid the costs of training the users of the systems [Plumtree G].

If the parts of the enterprise application that employees use more often were made more accessible and simpler to use, the time spent on these parts could be reduced considerably. Since these parts are used considerably more often than the other parts, or constitute the only parts used by many employees, the total time the corporation spends on this application as a whole will also be decreased considerably. This will free more time to do valuable work.

Another effect of making the application easier to use, is that the application may also be used more often, considering that the barriers for using it have been lowered. This can be advantageous; in many cases, the amount of data cannot ever be too much. An obvious example is the situation where the portal is used in an extranet scenario, where customers can place orders directly into the corporation's order processing application. Another effect is that the data entered may be of a better quality, as the users can access and "fix" missing or erroneous data in an instant. Finally, Plumtree [Plumtree G] points out that a corporation "… increases [their] return-on-assets, opening expensive enterprise applications to the broad audience for which they were originally deployed."

5.2.5 Knowledge Management

The terms information and *knowledge* go hand in hand; however, knowledge usually refers to something "deeper" than mere information. According to Merriam-Webster's Collegiate® Dictionary[6], knowledge is "the fact or condition of knowing something with familiarity gained through experience or association." Information is an essential tool from which to gain knowledge, but does not constitute knowledge in itself.

[5] EAI is not a new term. It usually refers to the integration, synchronization and communications between multiple enterprise applications.
[6] Merriam-Webster's is available online at http://www.m-w.com/. [Accessed June 3, 2002]

Knowledge management is therefore a somewhat fuzzy affair[7], but revolves around tools and techniques that help identify and map the intellectual assets existing in the corporation, spread and increase the knowledge throughout the corporation, and facilitate the individual in gaining knowledge. The term *corporate knowledge* in essence refers to the routines, methods and "know-how" the company as a whole has gained from projects and activities in the past, in addition to the sum of knowledge from each individual in the corporation. Since corporate portals are such a focal point of the organization, this would make a good place to put such knowledge facilitating applications and tools.

Knowledge Base

A *knowledge base* is a system where largely unstructured documents, articles and data are entered as *knowledge articles* along with *keywords* and often an *abstract* of the content. Users can query the system for information by a searching interface, usually rather advanced. One can often select which parts of the article that should emphasized during the search, e.g. title, keywords, abstract and full text search. Searching algorithms may also be "intelligent", utilizing various heuristics about the articles, for example by giving different weight to search term occurrences in the different parts of the article. Articles may also refer to, or link with, other articles, and new articles may supersede old articles. In this way, the use of keywords, categories and links ties the articles together and consequently embodies the structure that makes knowledge out of information [Grammer, 2000].

The 7 R's of Information Management

Butcher and Rowley [1998] attempts to define the knowledge processing within an organization with a conceptual cycle, shown in Figure 4. The figure is a composition of both the information cycle and the input and output for each stage in the cycle.

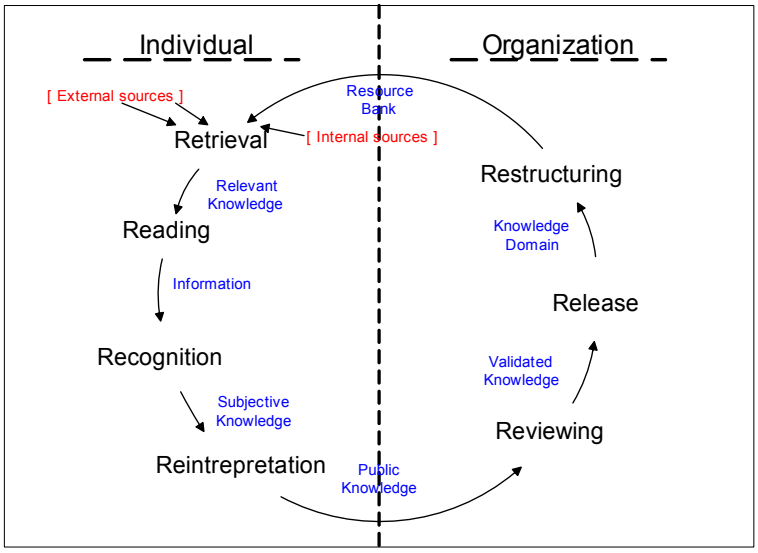


Figure 4 The Information Management Cycle ("The 7 R's of Information Management"). Blue text is the input/output for each stage of the process (Adapted from D. Buther and J. Rowley [1998])

[7] Resource site on knowledge management and related topics: http://www.brint.com/OrgLrng.htm

Several of these stages can be facilitated by tools in the portal. The search and *retrieval* of relevant knowledge is facilitated by searching functions in the knowledge base and other sources. *Reading* and *recognition* are purely cognitive phases, and can only be helped by the tools giving access to more relevant knowledge, facilitating the recognition phases. The *reinterpretation* is the process of converting the subjective knowledge into public knowledge, for example by writing a document. This can be assisted by effective word processing and document linkup facilities. The *reviewing* phase consists of validating the public knowledge. Knowledge systems can incorporate elements from the content management model to facilitate such reviewing. Content management is examined further in Chapter 6.3.2 – "Content and Document Management". Once validated, the knowledge can be *released* into the knowledge domain, for example by publication into the knowledge base. Finally, *restructuring* concerns the aggregation and processing of the knowledge to meet a specific purpose, leading to additional data that will complement the existing knowledge. In the knowledge base, this is facilitated by the linking and referrals between different information entities.

Expertise Localization

Portals can furthermore assist in the *localization of expertise* [PortalsCommunity, 2002]. A large corporation may have employees located all around the world. An employee might be at one location struggling with a difficult problem. At a different location, another employee might know the answer to this problem. Identifying such answers is often a complicated process, as some answers may be devised only by a person who has the necessary knowledge. Despite the readily available information, the answer may still be difficult to figure out.

Human resource management systems are used to char and measure each employee's knowledge and expertise fields. These systems enable users to look up anyone in the corporation who might have the knowledge necessary to solve the problem at hand. Informal and loosely structured information sharing systems like *discussion boards* enable the user to direct their questions to a relevant forum. By requiring every employee to subscribe to the lists that correspond with their knowledge, questions asked on the boards are channeled to the appropriate persons, who then can decide whether they can be of assistance.

Organizational Learning

Organizational learning refers to the process during which a company attempts to increase the corporate knowledge by increasing that of each employee, as well as to establish good routines and practices in general. Lessons learnt by mistakes must be recorded, as repeating errors are very costly and should be avoided. Brian Detlor [2000] argues that the *communication space* (described in the next section) facilitates organization learning. The reason is that both information distribution and information interpretation are important elements of the organizational learning process. When distributing information by making it easy for users to locate and consume information, the corporation as a whole will learn and gain new insights. When discussing and interpreting information originating in various sources, common understanding is achieved and new knowledge obtained. This new knowledge is fed back into the knowledge base and thus one more information-cycle is completed.

5.2.6 Collaboration

Collaboration is, according to Merriam-Webster's Collegiate® Dictionary, "to work jointly with others or together, especially in an intellectual endeavor". Collaboration applications are definitely not portal specific, *Groupware* or *Computer-Supported Cooperative Work (CSCW)* applications have been around for quite some time[8]. Considering the access to large amounts of corporate data, portals make an ideal place to work together. A portal is suited for project planning and project work, as the users may use the portal's information source to gather company documents and files. Each participant of an ongoing customer project may go in and update his assigned tasks and statuses. Uses may, by the use of a discussion system, post questions or answers and discuss the work with colleagues. Records of communications with the customers and clients can all be archived with the project. Moreover, it does not matter whether the colleagues are located across the hall, or at the other side of the globe.

Collaboration is an ambiguous term, and tools that facilitate collaboration can be anything that enables any form of explicit or implicit communication between several individuals. Several electronic collaboration systems might already be in use in an enterprise. Gates [2001] points out that today people do most collaboration using email. Other often-used collaboration systems include communication by chat and instant messaging, file sharing across distributed file systems and group calendar application. The portal's job is to make these forms more accessible and to unify the access to several of these applications. A portal might also have some of these applications embedded as portal modules.

Email

Email reading, writing and archiving can be embedded as a module. By accompanying this with an incoming email handling system, two benefits can be realized. First, incoming emails can be automatically routed, based on, for instance, the customer who sent it, and secondly, emails may be shared amongst several users. For example, an incoming email can be automatically linked to the correct customer, based on a match between the email address of the sender and one of the contact persons listed with the company. Simultaneously, the sales representative for that customer is notified of this incoming email by some notification or alarm system. If another sales representative, or the sales manager, checks up on this customer, the email will already be attached, and any replies that the first sales representative has sent are present.

Chat, Messages and Discussion Forums

In addition to the email system, which originally was meant for "long-haul" internet-distance messaging, three other communication systems or variants of emails are commonly used. These are *chat* or *instant messages*, *messages* and *discussion boards*. All of these are excellent candidates for integration into or implementation on a portal system [White, M., 2000]. A short description of each type of communication form follows.

- Chat is an interactive type of communication between two or more persons. It usually comes in one of two flavors. The first one is a very direct communication form, where each keystroke of the users is communicated to the other participant,

[8] Comprehensive resource page on CSCW and Groupware: http://www.usabilityfirst.com/groupware/

instantly appearing on his screen. This system appeared with the UNIX command "talk". The other type of system has lately been known as "instant messaging", and the concept is used in tools like Microsoft's Messenger and AOL's Instant Messenger. Using these systems, a user's message is not transmitted to the other parties until the enter-key is pressed. This is also a long-known communication form, popularized with the *Internet Relay Chat* system, IRC.

- Messages are very similar to email, except that they are delivered within the system itself. This means that there is no delay between the sending and reception of a message, making it somewhat similar to the instant messaging described above. The messages are kept in folders in the same way as emails. Notification of the reception of both messages makes fast responses to queries and questions possible. The portal system may have provisions for integrating messages and emails, in effect making the distinction disappear.

- Discussion forums or message boards are a messaging system where the message *inbox* is open to a group of users. The participants, who may be the whole community or a selected group of users, can all start a new *message thread* by *posting* a new message to the board, or can reply to an existing message, thereby doing a *follow-up* on the thread. Each board has some main topic of discussion, and posting *off-topic* messages is considered "bad netiquette". This type of system is yet another internet classic, known as the *USENET News System*, where each discussion forum has a hierarchical name denoting the topic of discussion and is known as *news groups*.

In addition to these very direct communication means, other more implicit forms have positive effect on the collaboration possibilities for a group of users. Some of these collaboration features is discussed below.

File Sharing

A *file sharing* system enables users to upload and download files from the portal server. This might be a specific upload/download procedure, requiring the user to download the file to some temporary location on his local disk, work on it, and then upload it. Some operating systems and applications have special provisions for working on documents located on a web server, notably Microsoft Windows running Microsoft's Office Suite[9]. The web server also needs some special software to handle this seamless integration. With these requirements in place, working with files from the portal server and working with files on the hard disk are very similar experiences, where only the actual "opening time" and "saving time" might differ.

The files shared can be located in a central repository, or can be attached to specific entities in the portal. A contract document with a customer for a specific transaction can be attached to the customer's representation in the *customer relationship management* part of the system, and the electronic copy of the user manual to some projector can be attached to the projector entity in the *resources* part of the portal. By having a proper taxonomy, one might be able to find a contract document both in the central directory, and attached to the customer entity.

[9] FrontPage (Office) Server Extensions: http://office.microsoft.com/Assistance/2000/FPSEreqs.aspx

File sharing systems is elaborated further on in Chapter 5.3.2 – "Publishing and Content, Document and File Management".

Group Calendar

A *group calendar* system will help the users organize and coordinate their daily activities. Most people are familiar with some kind of electronic calendar, which make it possible to enter, view and edit one's day-to-day *events* and *tasks.* Viewing is often possible in multiple fashions, for example *daily*, *weekly*, *monthly* and *annual view*. The purpose of the group calendar is to enable all the users of the system to share their calendars. Users are for example able to view co-workers' and superiors' calendars. This enables a user to consider other users' schedules while planning and coordinating their work. One can superimpose several users' schedules on top of each other, thus making it possible to find available "holes" in their combined schedule, or at least find spots where only a few of the people have something booked already. Similarly, one has the possibility to book meetings with a group of colleagues as attendees, submitting the meeting for approval directly to each of the members of the group. When all attendees have accepted the meeting, it is automatically added to their calendars, possibly notifying everybody of the schedule entry. A supervisor is often able to overrule a subordinate's calendar and may add or remove events and tasks in it without their approval, and merely let the system notify them of the newly entered event.

Resource Allocation

A group calendar application is often accompanied by a resource booking facility. A user that wants to arrange a meeting needs to allocate time from each of the attendees. Similarly, the user would also need to allocate a projector and a meeting room. Alternatively, an employee might need to borrow a company car for an off-site meeting. Such entities can be booked through such a resource booking system.

Brian Detlor [2000] argues for the notion of a *shared information work space* which can be formed within a portal system. This work space will, in Detlor's view, facilitate "… the creation, exchange, retention, and reuse of knowledge." This workspace is achieved by developing certain collaboration applications, as well as tools that enable information to be accessed, and workflow systems available within the portal.

Figure 5 is an information-based model of the corporate portal, and demonstrates the three major components that constitute the shared information work space. Chapter 5.3 – "Functional Components of the Corporate Portal" contains further descriptions of the applications and system necessary to facilitate these components.

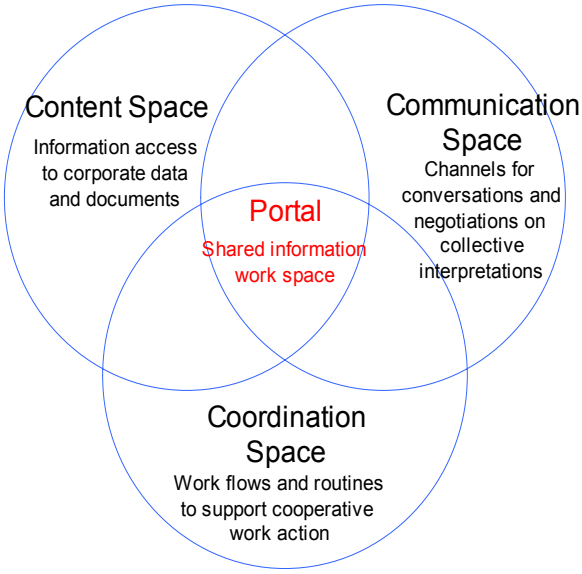


Figure 5 The corporate portal as a shared information work space (From B. Detlor [2000])

The *information content space* represents information storage and retrieval, and have been reviewed in the previous section. The *communication space* refers to the communication between the employees in the enterprise. Here collective interpretations and decisions can be made and meanings exchanged. This will help users interpret information, review it, and put new or refined information back into the system. Lastly, the *coordination space* is essential work flow systems and other work related routines which support cooperative actions, e.g. a shared project planning system. In addition, scheduling, organizing, negotiations, making ad hoc decisions, recovering from errors and assembling resources are actions that belong in the coordination space. Communications are crucial also in this space.

5.2.7 Remote Access

Being built on the open protocols HTTP, requiring nothing more than a web browser to access the system, the corporate portal enables *remote access*. Consultants, sales people and companies with several branch offices are likely to benefit immensely from this type of access, as accessing the corporate information system is just as easy from outside of the organization as from inside.

However, a frequently noticed adverse effect results from the lower bandwidth and higher latency that can be present on remote access. This negative effect is often insignificant, though, compared to the large upside such an access gives. One can also compensate the bandwidth problem to a certain degree by using so-called *low-bandwidth themes* when working from remote. These are themes where images and the layout are cropped down to a minimum.

In addition, security is a concern. The web protocol HTTP is not secured, meaning that all information is transmitted between the client and the server in clear-text. If a cracker can get physical access to any part of the communication path between the client and server, or manages to take remote control of any intermediary routing point, all information transmitted can be accessed without further effort. However, by employing the Secure Socket Layer (SSL), a high level of privacy is achieved. This

layer enables security by encrypting the information transport between the browser and the server. The resulting protocol, HTTP over SSL, is called HTTPS.

5.2.8 Integrate the Extended Enterprise

Usually, four communities are referred to when talking of the *extended enterprise*. First are the *employees*, who are all the people that constitute the corporation. The company's *customers* are the ones that pay for the goods or services, thereby keeping the company alive and thriving. Thirdly, the company usually depends on *suppliers* in one way or the other. Either the company needs products to resell, or it needs equipment and tools in order to deliver products and services. These three entities are usually referred to as the company's *value chain*. Lastly, a company often cooperates with *partners* in order to exchange products and favors.

Schroeder [1999] argues that, "a crucial feature of an enterprise portal is its ability to function on either side of the corporate firewall". By enabling the corporations to open their portals, administrative costs are reduced as the outside parties retrieves their own information. The free exchange of information is also vital to enable e-commerce.

Each community have their own information requirements, and charting these and making available the necessary information through the portal interface may help the corporation gain competitive advantage. PortalsCommunity [2002] has made a review of which information needs each group has and what might be gained from sharing this information. The following information is based upon this text.

Customers

The main reasons for deploying a customer centric portal solution would be to raise the company's ability to acquire, serve and retain customers. In the current marketplace, *competitive advantage* is becoming relatively more concerned with knowing the company's customers than with product features and innovation [PortalsCommunity, 2002]. Customer intimacy, relationships and service is paramount. BEA [BEA D, E] points out that customers these days demand that a corporation know who they are. The customers have had contact with the corporation several times, and repeatedly informed them of their different needs, problems and their ideas for solutions. The customers therefore expect that the next customer representative they make contact with should know all the necessary information concerning these earlier encounters.

In a customer focused portal, the users should be able to get information about products and prices, check current inventory, track current orders and delivery status, place new orders, view order and payment history, check payment statuses, place service requests and support calls. The corporation may track the customers' site searches and portal navigation and thus try to pinpoint ways to improve the site.

By offering such selected internal information, a corporation may gain several advantages. The customers may become so happy with the organization that *moving away* from it is hard. In addition, by being proactive and develop new and customer focused features, the company can gain a competitive head start. This is often called the *first mover advantage*. The corporation can also gain better marketing,

prospecting, sales, field service, customer relationship management, service and support, and thus get more sales.

Another advantage from direct customer interaction is that the corporation can get invaluable product feedback. The products and the relationship with the customer may be improved in directions highly driven by the customer's needs. Feedback forms, questionnaires and support forms might be inserted in strategic locations, making it easy for the user to inform the corporation of his experience with the products, service and customer relationships, including the portal. Given a good and quick response from the corporation, this will heighten the customer satisfaction and make it harder for them to leave the corporation.

Suppliers

The information needs for suppliers are in several ways the opposite from the customers. A corporation would want their suppliers to open up their information, so that it could check the suppliers' inventory and make proactive ordering decisions.

In several industries, typically the agricultural industry and other primary industries, the suppliers supply to only one or a few corporations. The corporation in question might not even be an independent organization in itself, but rather a co-operative for all the suppliers, for example an agricultural co-operative. In such settings, the suppliers are close to the employees group, demanding access to both extraction and entering of data. A farmer would for example want information on how many cows he has delivered so far this year and be able to order pick up for a new batch of chickens next week.

Partners

Corporations are becoming more focused on their core competencies. This makes a corporation dependent on a large and tight partner network from where to get the competencies that it lacks. A corporation must be able to enter partnerships fast and efficient. A good partner channel portal where a lot of information is being shared would enable both partners and corporate users to gather important information efficiently.

In order to increase sales, marketing documents, product data sheets and white papers should be distributed to the partners in an efficient manner. Sales leads can be shared amongst partners so that the best suited partner can take care of the customer relationship. Joint selling opportunities and strategies can be discussed and acted upon. Forecasts can furthermore be collected and analyzed, so that a more accurate picture of expected revenue can be formed.

Partners are integrating and reselling the corporation's products and services, thus gaining considerable insights in the features and values of the product. In addition, the partners' customers will also give product feedback and suggestions. The partners' issues and enhancement suggestion can thus be very valuable to the corporation, filtering and condensing their customers' feedback in addition to the partners' own experiences.

Technical information and training may be administered from the portal system. The corporation might have a knowledge base, in which they give the partners intimate

access. The partners may join the internal discussion lists and issues tracking system, and get to know product release schedules and given "insider" information.

Web Visitors

When the corporation has a presence on the internet, two more entities can be added to the concept of the extended enterprise: the *anonymous web visitors*, and the *registered web visitors*. These are people that are browsing the corporation's internet site, looking for some kind of information or simply checking out what the corporation has to offer.

Anonymous webvisitors are persons whom the web server cannot identify. It is often possible to follow such a user's moves around the website nonetheless[10], thereby getting an impression of what anonymous users find interesting and what information they are seeking. By watching variables such as which web page is more popular, page viewing time and number of clicks before localization, the site can be tailored and polished to suit the information needs of the persons that visit the site[11]. In addition, searching terms is valuable as they suggest what users are looking for, and which information is too hard to find.

Registered webvisitors have given the corporation information about themselves, most probably in exchange for some special resource. It is often easier to track these visitors than anonymous visitors, as their so-called *session* with the web server is well defined. In addition, repeated visits are easier to track, as the web server might both recognize their identification automatically, and because the user will give some credentials (typically some username and password selected by the user) in order to open up the special resources. These people probably have some tighter relationship with the company since they are willing to take the time to provide personal information. Analyzing their moves within site might give even more information about the corporation's site.

Both of these groups are either already customers which have not identified themselves yet, or may be viewed as *potential customers*. The corporation can tend to both, treating them as leads to possible sales.

Internet Strategy

A well-defined portal and internet strategy should tend to all these six communities. Some portal systems can handle all of this using the same system. This may give several benefits, as this one system can just dish out information and access to applications based on the access restrictions on the specified piece of information or application system in accordance with the defined role of the user.

[10] Most of the *sessionless tracking* schemes assume that a repeated visit from the same IP address within a predefined time (often 30 minutes), is done by the same person. This is not always correct, though, as multiple persons might be *masqueraded* as one IP address, for example by sitting behind a corporate firewall. Another way to track usage is by issuing a *cookie* to the user's browser (or by URL rewriting, which is a similar technique), thereby establish a web server *session* for the user. In this way, one can track precisely what each user is viewing.

[11] However, the needs and requirements of the customer and those of the business often vary. The user would like to get everything within one easily locatable click. The company might have another idea, as it would like to feed the web visitor as much information as possible before letting him to the information he really needs. In addition, some information might not be desirable to give to the user right away, e.g. contact information like telephone numbers; a company might prefer that web visitors should use some contact form on the website instead.

5.3 Functional Components of the Corporate Portal

Typically, several functional components are available from within a corporate portal system. This section will list and discuss some important elements that are common to many portals. The following list is mainly compiled from PortalsCommunity [2002], Colin White [2001] and McDonough [2001]. Eckerson [1999] has defined 15 requirements for a corporate portal, some of which are applicable here and have therefore been included. The components described here are of a generic nature, some of which have previously existed in the same or similar forms on different systems.

5.3.1 Taxonomy and Directory

A *taxonomy* is a structured view of all the corporation's documents and information. It is often set up as a *directory* with a *directory structure*, much like the file system of most operating system. The folders and subfolders hold different kinds of entries, though, and thereby work as a classification tree or hierarchy. The idea is to enable access to all information in an enterprise through browsing and searching this directory structure.

Some systems make use of *element placeholders* or *cards* (confer the Epicentric analysis). These contain references to the actual item, thus enabling any resource to be inserted into the taxonomy for categorization.

5.3.2 Publishing and Content, Document and File Management

Content is the information in the portal itself. Examples include articles, news bulletins, pictures and similar data that are displayed directly to the user, normally using HTML. It is different from *documents* in that content is usually displayed in the portal itself, while documents are files that may be downloaded to the local machine and viewed and worked on there. While some systems make use of a built-in or portal specific management system, other use and interface towards existing, third party systems, for example Interwoven and Documentum [Epi B, Plumtree B].

Content is usually managed and published into the portal using a *publishing system* [White, C. 2001]. This module is responsible for handling of everything that may be displayed within the portal. There is often a publishing cycle which begins with *authoring* and *contributing* a document. It is then *reviewed* and *commented* on by the author's peers or a dedicated *reviewer*. When the content is approved for publication into the portal, it is *approved* by a user with this capability. After approval, an *editor* will *publish* the content, often with a start and end date. The content might also be meant for just a group of users or for a set of users having a specific role (See below for a discussion of roles and groups), and the content will then have some access attributes set to it. The next time a user is watching the specific portal page, the new content will be delivered to the user by embedding it into the HTML output

The content can be created by some special tool, for example a dedicated HTML application, or there might be a content editor embedded directly in the portal. The latter approach is to prefer when it is desirable that everybody should be able to contribute and comment on the content.

Document and File management is very similar to content management, except that such systems handle documents and files instead of content meant for embedding in the portal's pages. The publishing cycle may still be used, as e.g. a product data sheet

will have to be accepted by the sales manager or product manager before a sales person have access to it to and can give it to a customer. Document and file management often make use of check-out and check-in procedures. This is meant to prevent several users to simultaneously download a document, edit it, and then upload it, thereby loosing one of the users' changes [Shilakes and Tylman, 1998].

In addition, management systems like this can easily handle *versioning* and *change logs*. By comparing the newly uploaded file with the existing one, the system can se if it really changed. If it has changed, a copy of the old file, or the changes between the old and the new file, is stored in some kind of archive, thereby enabling users to track the changes for each revision, and to fetch old copies of the file.

5.3.3 Searching

The search function of a portal system is a very important tool. The searching system is indexing content and documents in the portal and from different sources across the enterprise, enabling retrieval of data and documents by the user entering some search criteria. Specifically, the content of the taxonomy is searched, and the results might send the users to the correct node in the taxonomy, thereby giving them a relevant context from which to drill further down into the data. Some systems make use of an external, third party search engine. For example, the Epicentric system uses the Inktomi Enterprise Search Engine, while Plumtree uses the Verity search engine.

The searching system might be facilitated by a so-called *crawler.* This is a program that runs through the company's existing files and documents, indexing the content and metadata about each file. This enables a full search through all information in the company through one single search interface.

Plumtree have introduced the concept of *network searching*, called *federated searching* by Verity. The idea is that by assembling a single search criterion, the user is enabled to search through multiple data sources.

5.3.4 PDA integration and synchronization

It is still difficult to bring along internet access and a laptop everywhere. This makes the Personal Digital Assistant indispensable to many business functions, keeping track of the owner's meetings, tasks and contacts. The problem is that this will be yet another data registry, with some meetings registered in the PDA and some in the portal's calendar. This is where the *synchronization* enters as a viable alternative, transferring the data back and forth between the PDA and the portal's database. Changes in the PDA are copied to the portal, and vice versa. Different changes to the same record, for example that a customer's address is changed on the PDA, wile the customer is deleted on the server, will have to be resolved manually.

A growing number of PDAs are becoming mobile networking appliances, with internet access either directly embedded or via a mobile telephone. This enables a PDA to access the portal and its data directly, online. Enabling access from PDAs can be achieved in different manners. Applications can be built for the PDA that accesses the portal's data directly, thereby enabling the access of certain data from a PDA. Another way to integrate is at a higher level, where the view of the data is tailored for some other viewing application other than a HTML based web browser, e.g. the more minimalist WML browser on WAP capable devices. At the highest level, several new,

advanced PDAs have full-blown web browsers, enabling the PDA to directly access the portal in the same manner as one would at a stationary computer. To facilitate the usage of the portal from the reduced screen sizes which the PDA offer, it would be appropriate to make a special theme, or skin, tailored for the PDAs' screen sizes.

5.3.5 Workflow and Business Process Automation

Many operations in an organization require several steps for completion. Let us take an example of hiring a new employee. This new employee will need a new user in the computer systems, an order for a parking lot space must be fulfilled, a new employee access card must be made, the wages and benefits system must be updated and several operations like these must be fulfilled. All of them must be properly done, and some of them cannot be done until some other task already are finished, making a dependency tree of task. Several of the tasks are done by different people in different parts of the organization. This flow of operations is commonly called a *workflow*.

This whole operation may be made more efficient if the workflow is supported by some system. This is *business process automation*. The manager could start the process by initiating the predefined operation "new employee". The portal then presents a form in which the manager must type the personal information of the employee. The system will have several required fields, and not let the manager pass on the flow before all these fields are filled. When everything is filled out properly, the manager hits some "initiate new employee workflow". A task is then submitted to the next persons in the flow, parallelizing all the tasks that can be done simultaneously. Operations where some information may be incorrect can be sent back to the person which is responsible for that part, e.g. when the accountants' department tries to fill in the social security number for the employee, the accounting system find the information incorrect. The form can then be sent back to the manager with a note of the problem. When the accountants' department has done their task, they hit a "next in flow" button, sending out new notifications to the persons which tasks are next in line. When all the operations are completed, the manager might get a final notification which informs him of the completion of the "new employee" workflow.

A portal system is very well suited for having good workflow systems embedded in them, since the portal spans the entire corporation and by design and intent integrates towards several of the underlying systems that must be accessed to fulfill each task.

5.3.6 Alerts and Notifications

Alerts and notifications are similar concepts, often using the same mechanism in the portal. An alert is sent when there is a need to notify the user of some condition that has been met, e.g. that there are 15 minutes left to the scheduled meeting, or that a deadline is approaching. A notification is sent when some event happens, for example if the user receives an email or some new task appears in the user's workflow queue. Both concepts require the user to be notified in some manner.

The alerts and notification system can be built into the portal system, or available as a service to the portlets. Several ways of notifying the user can be available to the portlet; each used in different settings or attached to different notification levels. There might be an alert-portlet, showing the different alerts and notifications that have occurred. Then, there might be a mechanism of sending an email to the user so that he

will see the notification in his email inbox. As an even more urgent notification, paging the user or sending a text message to his or her mobile telephone can be possible.

5.3.7 Push Information and Subscriptions

Several articles, including the original analysis from Shilakes and Tylman [1998], mention the possibility to "push" information to the user, instead of requiring the user to "pull" it off the system. This is similar to the concept of notification, where certain events are programmed or configured to send out a notification to the user.

This is also called subscriptions, since the user is subscribing to certain types of information. When a new sales document is added to the sales document repository, the rest of the sales organization would probably benefit from knowing this. The administrator could configure the portal system to send a notification or send the entire document in an email to all the sales representatives in the company.

5.3.8 Users, Groups, Permissions and Roles

By providing easy and direct access to vital company data, powerful, yet simple to employ security mechanism must be available [White, M. 2000]. Several vendors employ two different concepts to control access to the information and functions in the portal system.

Users, Groups and Access Control Lists

The concept of access control lists, or ACLs, are embedded in some form in most operating systems. The idea is that every object in the system has attached a list of users that are granted different accesses to the object. Access rights are different in different context, but may include the "standard" rights like read, write, change and delete an element or object. However, in other contexts, rights like grant, access and customize might be more appropriate.

A user group is, obviously, a set of users. A user group is often used to simplify access control or restrictions. Instead of individually granting each user a right to some object, the users can be put into a group. This group can later be granted rights, and each member of the group gets this right.

Roles and Permissions

While rights are used to control access to different objects in the system, permissions, also called capabilities, are used to control access to different functions within the system's applications. Roles are assigned to users, and a user might have several roles. For example, a portal system could be configured to have, amongst others, the roles "sales manager", "sales representative" and "employee". All employees in the corporation are assigned the employee-role, while all employees that functions as sales representative are given the sales representative role, and finally are all employees that functions as sales managers given the sales manager role.

Further, a role is constructed by putting together a set of capabilities. To continue the example: a sales application might have, amongst others, the permissions "authorize contracts", "create contract" and "view contracts". These permissions are assigned, by the portal administrator, to the different roles mentioned above. The role "sales manager" is assigned all these permissions, "sales representative" is assigned the two

last ones, while the role "employee" is assigned the single permission "view contracts".

These two different concepts, ACLs and permissions/roles, overlap to some degree. In several ways, it refers to two different approaches to restricting access. From one side, a type of objects can require a specific permission to be worked with, thus implicitly granting a user access based on this user having the specific permission. On the other side, the user can instead explicitly be granted the access via the object's ACL. However, the ACL approach provides for finer grained control, as each object has its own ACL. Together, a user group and a role and capabilities system form a strong basis to control and restrict individual users' access to data and functions of applications within the portal. In addition, properties of the user experience, e.g. theme selection and internationalization, might be pre-selected or locked down for a group of users, or for a set of users having the same role.

5.3.9 Single Sign-On

One major appliance of portals is *Enterprise Application Integration*, where information and selected features from underlying sources and applications are made available through the portal. Most of these underlying systems require authentication of some sort, usually, still, in the form of a username and password. The portlets must thus impersonate users, connecting to the underlying systems on behalf of the logged on user. This would either require the user to supply the necessary credentials for each operation, or let the portal store these credentials for the user, applying some single sign-on mechanism.

Sometimes the company already has some main authentication mechanism in place, e.g. a Kerberos system or a LDAP source. The users' portal access can then be authenticated towards this same source. Most often, one or several of the company's data handling applications are also using this same authentication mechanism. The same credentials used for the portal logon can therefore be used to log on to the underlying applications and systems.

If the credentials for portal access and the underlying systems' access are different, the portal can keep a repository for each user's credentials for each application that will be accessed. When a user action inside the portal requires some access to the underlying application, the application connects to the source via the portal's credentials repository. The portal provides the system's authentication mechanism with the appropriate credentials and thereby simulates the actual user. This facility is provided by the IBM framework.

5.4 Portal Categorizations

There are several categorizations of portal systems, and many of the categories overlap considerably. Martin White [2002] comments that there seems to be almost as many categorizations as there are consulting companies. The term "portal" has been widely used and abused, and there is currently a sizable set of different ideas of what the portal is. As this thesis is about the concept of a *corporate portal*, which is quite clearly defined by Shilakes and Tylman, portal segments like the *internet portals* (alternatively *web* and *public portals*, e.g. MyYahoo and Yahoo! itself), *vertical portals* (Business-to-business, B2B, e.g. VerticalNet @ http://www.verticalnet.com), and *e-commerce portals* (Business-to-consumer, B2C, e.g. Amazon @

http://www.amazon.com) are left out. For a review of these categories, consult PortalsCommunity's [2002] chapter "Portal Definitions and Types of Portals".

Following is a categorization that will shed some light on the selection of frameworks done in the following vendor analysis. This is the "coming from where" categorization, which divides the different vendors into five groups depending on which business area they are coming from. The categorization is adapted from Martin White [2000].

Collaborative vendors

These vendors have a strong background in the collaboration or group ware and document management sector. Typical examples are Autonomy with Autonomy Portal-in-a-Box, Lotus with Lotus K-Station and Verity with Verity Portal One.

Business Intelligence vendors

These companies originate in the business intelligence, reporting and data warehouse and marts sector. Typical examples are Brio with Brio Portal, Business Objects with Business Objects BI Portal, Cognos with Upfront Portal and Hummingbird with Hummingbird EIP.

Enterprise Resource Planning

Coming from the Enterprise Resource Planning sector, several of the sector's leading actors have established a portal product. Typical examples are SAP with MySAP, PeopleSoft with PeopleSoft PortalSolutions and Baan with iBaan Portal.

Existing vendors, new market opportunities

Several of the major IT and software companies are viewing the corporate portal concept as a new sales opportunity, making frameworks which adapt their existing technology to the needs of the corporate portal paradigm. Some examples are Oracle with Oracle 9iAS Portal, IBM with WebSphere Portal, Microsoft with SharePoint Portal and Sybase with Sybase Enterprise Portal.

New entrants

In addition to these existing companies, new entrants have emerged with this new market. They are mostly delivering frameworks instead of pre-built solutions. Some representatives are Plumtree with Plumtree Corporate Portal, Epicentric with Epicentric Foundation Server, Viador with Viador E-Portal, InfoImage with InfoImage Portal, and Apache with the open source system Jetspeed EIP.

Companies with an existing background, such as in the three first categories, are essentially making their already existing applications available via the web browser, utilizing the layout and making personalization features available with the portal paradigm. Their products are often not made to be extended, and they are "monolithic" in structure. On the other hand, they provide a good set of features, as they all have a significant history in their respective sector.

The fourth category also has a background, but is providing frameworks for building portals, or interfacing with other portal frameworks. Dependency towards the vendor's own solutions is the rule, not the exception, in this category. That is, Microsoft's EIP-like SharePoint Portal system is heavily dependent on other

Microsoft products, essentially merging several of its own technologies into a web-enabled system.

The last category consists of independent vendors that have developed a stand-alone system that integrates multiple underlying sources. Most often, however, these solutions need some system on which to run. A full Java 2 Enterprise Edition application server or the lesser Java Servlets container is often employed. As these vendors do not have any existing applications, they are starting out from scratch. That means that these frameworks may come without any features or portlets at all, expect from a user management tool and similar administration specific tools. Several vendors nevertheless have a library of already existing portlets and modules, confer the analyses of Epicentric and Plumtree. These modules may be used to quickly assemble some base functionality in the portal. After that, solution specific portlets for the company in question can be developed in a modular fashion, expanding the functionality as the user needs emerge.

This thesis is primarily considering these two latter types of frameworks, and the subject is thoroughly covered in the next chapter.

6 Corporate Portal Framework Vendor Analysis

The following chapters consist of a review of what functionality some prominent modular development frameworks for corporate portals are giving today. The analysis is carried out by doing an analysis of selected corporate portal frameworks, followed by a summarization and discussion of these frameworks' features and concepts.

6.1 The Need for a Portal Framework

The need for a corporate portal development framework is stressed by Colin White [2001]. By supporting a combination of the *bottom-up* and *top-down* approaches for developing information technology solutions, he argues that a *federated portal framework* gives the best combination of flexibility and totality.

Bottom-up development implies that each corporate unit, for example each group or department, may develop their own portal solutions, tending to their specific needs. This gives great flexibility, and ideas and issues can be tended to quickly. However, multiple installations lead to increased administrative costs and overhead. In addition, the corporation's global needs are difficult to tend to, as the corporation's employees are scattered about on their different portal systems.

The top-down approach is at the other extreme, where developers surveys every need in the corporation, and then develops one large, monolithic solution. This gives a system that can take the corporation as a whole into consideration, and synergies from an enterprise-wide collaboration platform can be reaped. However, this is an inflexible system, where changes are very difficult to implement and new ideas cannot easily be implemented.

By having a system that enables developers to plug needed functionality easily into a common portal platform, the corporation is enabled to fix business "pain-points" rapidly, and thus achieve a quick return-of-investments. At the same time, these components should be easily integrated into the corporation's total system, giving long-term gains for the whole organization. This federated portal framework should provide for all the functional components mentioned in Chapter 5.3 – "Functional Components of the Corporate Portal", or provide programming environments for developing such functionality easily.

Luce [2002] states that the expectations for portal platforms currently are being raised dramatically, "and portal platforms must now offer comprehensive architectures embracing all of the key areas of portal functionality: integration, categorization, search, publishing and distribution, process, collaboration, personalization and presentation."

6.2 Vendor Selection

Most of the companies included in IntranetFocus' [2002] comprehensive list of portal vendors are providing packaged solutions where the portal comes as a complete packaged product, or are merely providing a "portalization" of their existing products. Using the categorization described in Chapter 5.4 – "Portal Categorizations", only the two last categorizations are deemed to be of interest, as these companies' main

intention is to provide frameworks for the development of corporate portals. These are the "Existing vendors, new market opportunities" and "New entrants" categories.

Based on this bias from the categorization and input from CoreTrek, a set of vendors have been selected for the analysis. These are listed in Table 2. Included are two of the new entrants, dedicated specifically to portal systems, a couple of the "large names" in the software industry, and the open source framework Jetspeed from the Apache Software Foundation.

Apache Software Foundation	
Product	Jakarta Jetspeed EIP
Sole product	No, Apache have a large set of products, both based on the Java platform and other programming languages. Most known for the Apache Web Server. Apache is an Open Source community
Type of documents	• Few high-level, overview and architectural documents • JavaDoc API documentation • All source code files
Installed	Yes, version 1.3a2 binary distribution

BEA	
Product	BEA WebLogic Portal
Sole product	No, application infrastructure including one of the leading J2EE application servers
Type of documents	• Very good documentation, both high-level overview and low-level details • JavaDoc API documentation • All JSP files that are used to run the server are available
Installed	Yes, WebLogic Portal 4.0 on WebLogic Server 6.1 SP2

Epicentric	
Product	Epicentric Foundation Server
Sole product	Yes
Type of documents	• high-level, sales-pitched type of documents • Architectural overview documents
Installed	No, only available to partners

IBM	
Product	WebSphere Portal
Sole product	No, full range of hardware and software
Type of documents	• Sales-pitching documents • Portlet Development Guide • JavaDoc API documentation available for the older version, 2.1
Installed	No, not available yet (Version 4.1). Due to be released June

Oracle	
Product	Oracle 9iAS Portal
Sole product	No, Oracle is widely known for their extensive database solution and several enterprise applications built on this.
Type of documents	• Good documentation, both high-level overview and low-level details • JavaDoc API documentation
Installed	No.

Plumtree	
Product	Plumtree Corporate Portal
Sole product	Yes
Type of documents	• high-level, sales-pitched type of documents • Architectural overview documents
Installed	No, only available to partners

Table 2 The Selected Portal Framework Vendors

6.3 Analysis

Many of the scientific papers and articles are referring to the different vendors' portal systems, referencing and giving brief explanations to some of their technologies. In addition, several analysts have published articles where summaries of a selection of vendors are given. The technology consultant company Owendo [2002] have published a review entitled "The Lowdown on Enterprise Information Portals (EIP)", in which they give "brief outlines" of 13 different vendors. An article from John K. Waters [2000] titled "Portal Wars" also have very short outlines of 30 different vendors. These articles have given some start-information of what to expect from the vendors.

Documentation

All relevant documentation that was available from the selected vendors' websites at the time of writing has been downloaded for the analyses. Places searched were both in the public part of the sites, and the sections available by registering as a user. This includes so-called white papers, data-sheets, technical overviews, programming tutorials and similar documents. It is worth to notice that these companies' websites are very dynamic in nature. Documents that are available one week may be deprecated the week later. In addition, some site, for example Sun's Java site, operates with session-based downloads, thus rendering the URLs meaningless. It is therefore often difficult to give a proper URL reference pointing to a specific document. Appendix C, References, contains the title of the document, and, if the URL is too difficult to handle, the filename. Single web pages are not treated as documents as such, and are merely referenced as footnotes throughout the analysis sections.

Methodology

The collected documents and information were studied, and the architecture, architectural features, framework features and framework concepts have been summarized for each vendor. The *software quality model* from ISO/IEC [ISO/IEC 9261-1] has been used during the analysis, specifically the capability-definitions listed

in Appendix B. This thesis' definition of features and concepts does not match directly to any of the capability definitions in the software quality model; hence, it is used mostly as a guide to what to search for. However, if a vendor has any "specialty" that may be described with one of the capability-definitions, this is specified for that framework. For example, if a framework had any special provisions to enable the developers to test their new portlets, this is noted under the "testability" capability. In addition, if a framework has any obvious negative features, this is also noted throughout.

As this thesis is meant to be a literature review, the analyses mainly relies upon the vendors own documents. This is not deemed problematic, as the intention is to get a general overview of features and concepts by exploring the ideas of other vendors of such frameworks, and not to arrest the vendors for over-advertising their products. In addition, the Epicentric and Plumtree system is not available for developer or evaluation download. IBM's system is not available until June this year. The Oracle system is available for download, however, the sheer size of the installation set (three CDs for the database, and an additional three CDs for the application server) combined with the seemingly elaborate installation procedures made the installation difficult, and it was thus decided against. However, the Apache Jetspeed and BEA's systems were available for download, and was thus installed and tested.

Several of the vendors' documents are of a very basic nature, or intended for decision makers rather than software engineers. Often, there is a lack of proper API documentation and similar low-level information about the frameworks' programming environments. This can potentially give the effect of missing some interesting features or concepts. Table 2 includes a description of what kind of documentation that was found on the different vendors' web sites.

Java 2 Enterprise Edition

As it turned out, most of the portal systems evaluated need some or the entire Java 2 Enterprise Edition environment to run on. Some understanding of this standard will be essential for the following chapters. A review is given in Appendix B.

Further division of the thesis

Chapter 7 contains one subchapter for each vendor analyzed. The subchapters are further divided into sections for architecture, framework concepts and framework features, and a section for references to the software quality model.

Chapter 8, 9, 10 and 11 summarizes and discusses the findings from the preceding chapter. It is worth to notice that some elements are mentioned in more than one chapter, as they are referred to in different contexts.

Chapter 12 tries to deduce some probable paths of evolvement of these systems, based on current trends in the software market in general, and the portal market in particular.

7 The Vendors

7.1 Apache Software Foundation – Jakarta Jetspeed

Apache Software Foundation, ASF, is a non-profit corporation, formed to help the development of open source software[12]. The ASF shall provide a foundation for open, collaborative software development projects by providing hardware, communications and infrastructure. The ASF is a legal entity so that companies and individuals can donate resources while be assured that those resources will be used for the public benefit. It shall protect individual developers from legal suits directed at the Foundation's projects, and protect the Apache brand. The ASF was formed out of The Apache Group, which was a group of individuals that initially formed in 1995 to develop the Apache web server.

The ASF currently hosts several projects. Each project's members consist of a loosely knitted group of software developers. All the ASF's software is Open Source, licensed with one of the least restrictive open source licenses in use. The ASF is definitely most known for their web server, which is the most used web server in the world[13]. This web server was previously known simply as "Apache"[14], but this term is now reserved for the software foundation itself, and the web server has been renamed to Apache Web Server, or commonly Apache httpd[15].

The Jakarta Project is one of ASF's projects. This project's mission is "… the creation and maintenance of commercial-quality, open-source, server-side solutions for the Java Platform, based on software licensed to the Foundation, for distribution at no charge to the public."[16] Their most visible product is the *Tomcat* set of Servlet containers, although Jakarta has several other popular products. Tomcat 3.x and Tomcat 4.0 are in fact the reference implementation for the Java Servlets specification 2.2 and 2.3, respectively.

A subproject beneath the Jakarta project is the *Jetspeed Enterprise Information Portal*. The project started in 1999[17]. The scope soon expanded somewhat, and it became apparent that the system was heading towards being a general engine for web applications. Discussions on mail lists arose, and the *Turbine*[18] project was spawned. Turbine is now a web application framework that is being used by several projects, including Jetspeed.

---

[12] http://www.apache.org/foundation/faq.html
[13] Netcraft perform a monthly study of the different vendors' penetration in the web server market. Their April 2002 study included more than 37 million sites, and showed that over 56% is using the Apache Web Server. http://www.netcraft.com/survey
[14] As so many other names and terms in the software industry, the name Apache server also has its history. The name stems from the phrase "A patchy server"!
[15] "httpd" is a unix-ism, where many server processes, called "deamons", have names ending with "d". HTTP is the protocol, and the server process serving these requests thus gets the name httpd. Other examples are nfsd, fingerd, talkd, ftpd, ntpd and sshd.
[16] http://jakarta.apache.org/site/mission.html
[17] http://jakarta.apache.org/jetspeed/site/index.html
[18] http://jakarta.apache.org/turbine/index.html

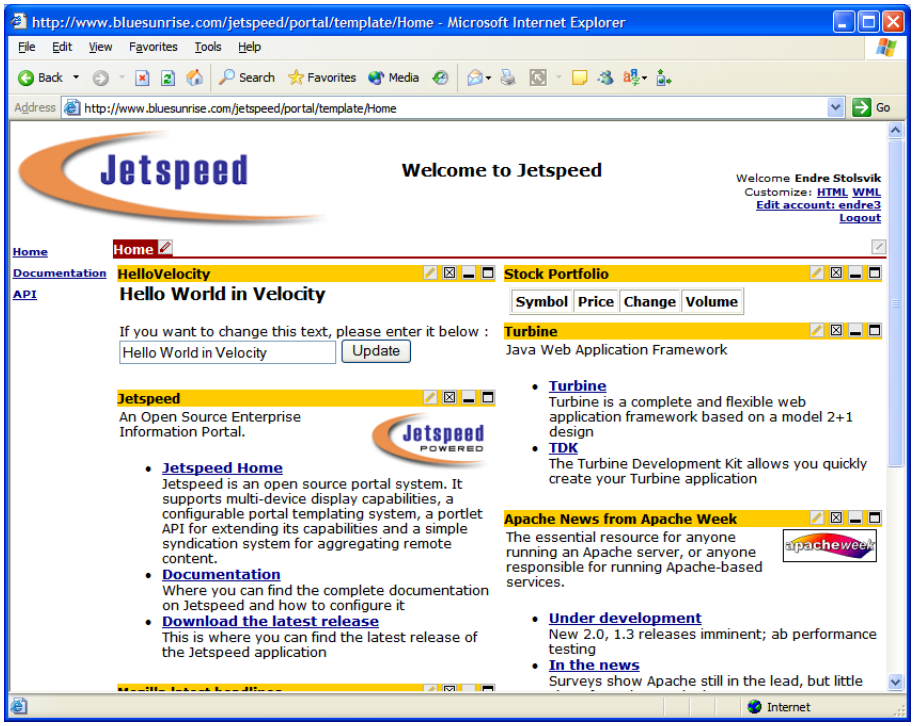


Figure 6 Screenshot of a new user's view of Jetspeed EIP (from demo site @ Bluesunrise)

The focus for the review will be the Jetspeed framework, while Turbine's features will be examined when necessary for the understanding of Jetspeed. The Jetspeed EIP project is at version 1.3a3 now, i.e., in an alpha stadium of development. Some time ago, it was released as version 1.0 and later 1.1, but at some point, the developers rethought their decisions, and concluded that Jetspeed was not mature enough for production release. The 1.2 version was therefore bumped to 1.3 and retracted to an alpha status[19]. Amongst the current topics on the Jetspeed mailing list, the first beta release of Jetspeed is discussed, and is apparently soon to be released.

Testing shows that the Jetspeed system clearly is not suited for production environments. The author's installation went smoothly (Only a 1.3a2 distribution were available at the website, and the CVS checkout did not compile), but several of the features seem to be in a very early stage. Some features are just non-functional, while other goes awry or crashes when used.

In addition, the documentation is, like with many open source projects, scarce. Developers are in general more interested in writing more code and new functionality than writing documentation. This comes in addition to the fact that the developer sees the whole system as perfectly logical and easy to grasp, while newcomers might have substantial problems with the extensiveness of it all. Without any financial resources, it is difficult to get someone to write proper documentation. Jetspeed suffers from a serious lack of overview documents that explains how the different levels beneath works, how the different entities connect and what the lifecycles and call-patterns for the different entities are. However, the source is available, but this is very low-level, making the high-level overview understanding hard to achieve. These matters have made the review process for the Apache Jetspeed portal time consuming.

[19] Jetspeed FAQ: http://jakarta.apache.org/jetspeed/site/faq.html

However, there are several reasons why the project still is interesting to review. The feature set of the platform is rich, and the system's portlet API and associated services are interesting. The system is open source and free, which alone are interesting properties. Lastly, the ASF is a member of the Expert Group currently designing the Java Specification "Portlet API". The Apache's next generation portlet API, which were developed in conjunction with IBM, will most likely affect the final Java standard Portlet API to a considerable degree. Apache and IBM's systems are therefore interesting concerning which functionality will be included in this standard.

7.1.1 Architecture and Architectural Features

Jetspeed is, as mentioned, built upon the Turbine web application development framework. Turbine is built on Java Servlets. Jetspeed implements apache's current portlet API, and on this API the developer programs portlets. The resulting technology stack is pictured in Figure 7.

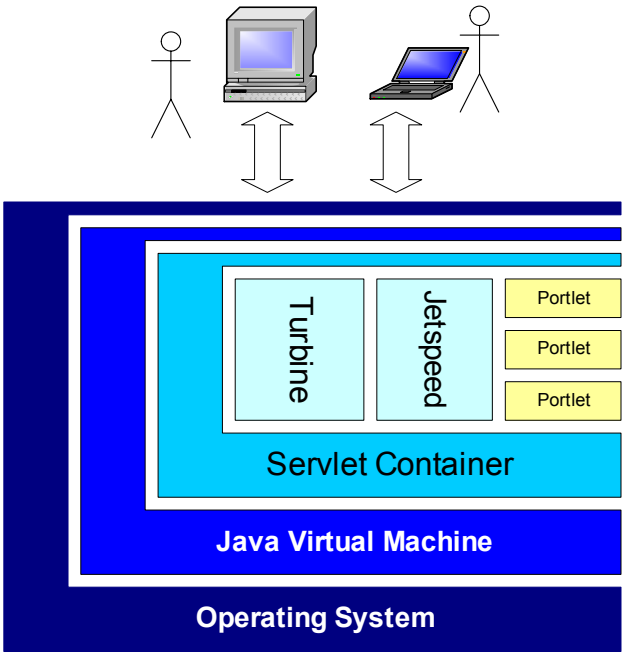


Figure 7 Jetspeed Technology stack

The Jetspeed system needs a base database. This is used to store user authentication and customization data. The package ships with Hypersonic SQL, a Java based RDBMS[20]. This enables a quick test-installation.

Turbine

Turbine is a framework for developing Java based web applications. It is filled with features and services specially designed to make the development of web applications easier. Turbine may be used as a complete Servlets based development framework, by using the Turbine Servlet as the main controller. However, a developer may choose to use only some of Turbine's set of features and services, thus utilizing Turbine as a repository of useful code. Jetspeed is built on top of the Turbine Servlet.

[20] The database HSQLDB, including Hypersonic SQL: http://sourceforge.net/projects/hsqldb

Turbine takes care of the authentication of the user. It defaults to use application level authentication, thereby circumventing the servlet container's authentication procedures. The documentation argues that the problem with container-managed security is that it cannot easily be managed from within the application[21]. However, Turbine does also support container-managed security.

The system consists of an overall model that includes Actions, Layouts, Navigators, Screens and Pages[22]. This is pictured in Figure 8.

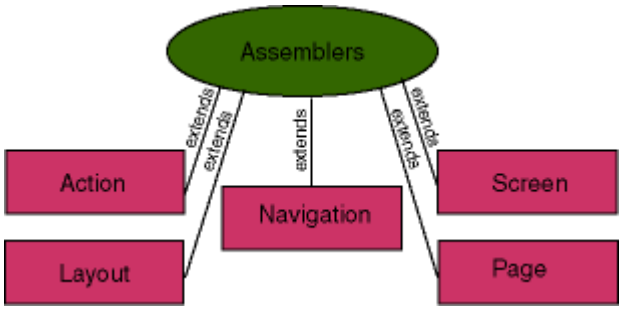


Figure 8 The entities in the Turbine Specification (from Turbine's website)

Actions are user (browser) initiated actions. The rest of the entities are concerned with the creation of the web page's content, and their relationship is shown in Figure 9.

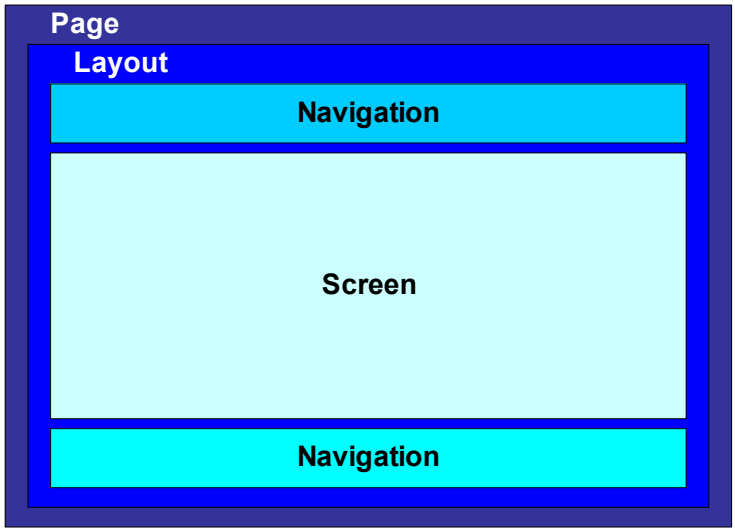


Figure 9 The relationship between the Page, Layout, Navigation and Screen entities (Adapted from Turbine's website)

When a request comes in from the browser, the turbine framework checks whether the user is logged in. If not, the login page is shown. If yes, the chain of events are as follows:

[21] Turbine J2EE integration: http://jakarta.apache.org/turbine/turbine-2/j2ee-integration.html
[22] Turbine Specification: http://jakarta.apache.org/turbine/turbine-2/fsd.html

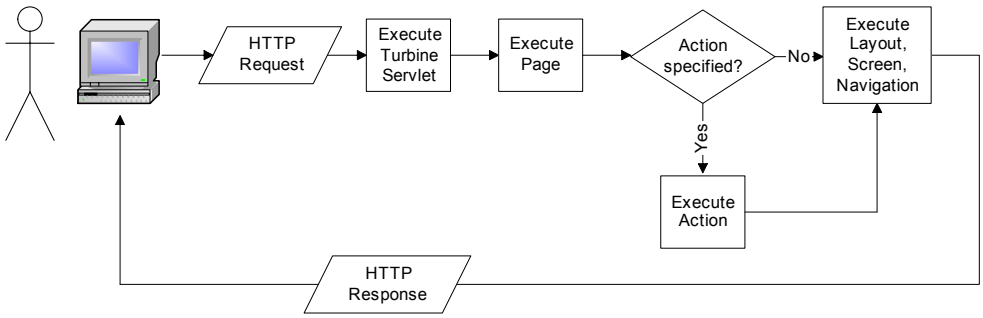


Figure 10 Turbine event chain (Adapted from Turbine's website)

The users request includes which screen the application should perform and what action should be performed. If an action deems it necessary, the screen that will be shown may be altered before passing on control. After the action is invoked, the resulting screen is requested for which layout should be used. The returned layout is executed, which in turn executes and includes the content of the navigations and the screen.

The rendering of the different parts of the finished page are done using a rendering system, currently supported are WebMacro, Velocity, Freemarker and JSP.

Services and functionality from the Turbine framework include database connection pooling, HTTP parameters parsing, event based action handling, job scheduler, access control system, role based security model and integration with several database object-relational tools, including Turbine's own; Torque.

Jetspeed

Jetspeed is implemented as a web application on top of Turbine. It uses much of Turbine's functionality, and includes some portal-specific functionality as well. Depending on the chosen template language, Jetspeed uses one of three Turbine pages. The view is assembled by executing the different parts of the page, as explained in the previous subsection. Three navigation bars are in use in the default Jetspeed installation: top, bottom and left. The screen is executed, which in turn executes the correct rendering template, called Controller, based on the user's chosen portlet layout. This template in turn executes the different portlets residing in the controller's PortletSet, by invoking them through a Control template that outputs the portlet frame and control buttons like close and minimize. The control template finally includes the portlet content. This system is explained in greater detail in the subsection "Portlet API".

There are some features that seems misplaced as to in which of the two frameworks they are placed. For example, there is a concept of a *capability-map* that maps which capabilities the connected browser have (explained in the next subsection). This functionality is implemented in Jetspeed, and not in Turbine as one would expect based on the generality of the function. The same goes for the different markup supports, e.g. WML and HTML, which also is implemented in Jetspeed.

7.1.2 Framework concepts

Panes

The workbenches in this framework are called panes. The system allows the user to make new panes. A pane uses a *Controller* to arrange the content. The controllers are called Layouts when shown to the user. Interestingly, there are two types of layouts; the first one can contain portlets, while the second one may contain a new set of panes.

The configuration screen enables the user to add portlets onto a portlet-layout, and panes to the pane-layouts. There are two pane layouts available in the default configuration. The first one is called “Menu pane”, where the added panes goes into a list on the left side of the containing pane. The other one is called “Tab pane”, and gives a new horizontally tabbed pane-set. Within one pane, one may choose to use yet another pane-layout, thus making a hierarchy of panes. Figure 6 only displays one level of panes, and this set of panes only contains one pane, “Home”.

There are several portlet-layouts with different numbers of columns, and different widths of the columns. There is for example “Single column”, “Single Row”, “Two columns (25/75)” and “Three columns (25/50/25)”. One apparent problem is that there is no correlation between the sizes of the portlets and the column-width. This means that if the user place a very wide portlet into a narrow column, the rendered page will potentially be much wider than the browser window, in effect ruining the workbench.

Multi-markup and Capabilities

Jetspeed is multi-markup language aware, and can output its content in both HTML and WML. There is a concept of a capability map associated with the connected client. This concerns both which mime types the browser supports, and which capabilities it has. Capabilities include whether it can handle frames, forms, images, javascript, tables and so on.

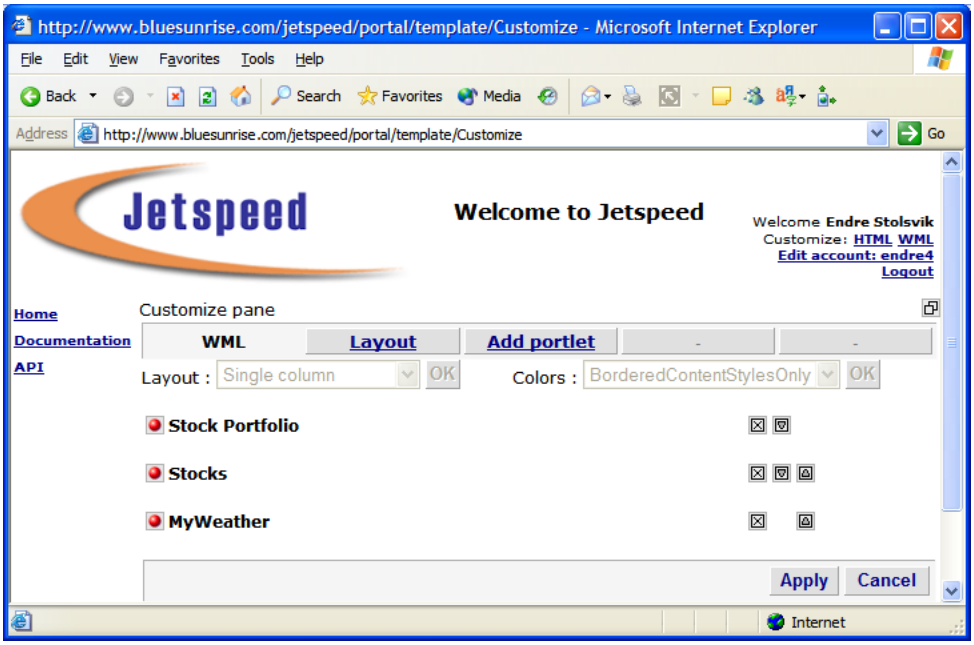


Figure 11 WML customization. Notice the two customization options in the top right corner, HTML and WML.

The stock Jetspeed distribution includes support for HTML and WML. As can be seen from Figure 11, both the WML customization screen's layout and colors selection drop-downs are grayed-out.

The different markup languages are configurable, and adding a new language is easy. The portlets must both be programmed to handle multi-output, and configured so that the framework knows which portlets handle the different markup languages. In the test installation, only the three portlets shown in Figure 11 supports WML.

Roles and Permissions

Turbine uses a permissions based role system. One or more roles are associated to a user. A role consists of one or more permissions. A developer may query the *AccessControlList* object for whether the user has a specific permission and whether the user is assigned a specific role.

Omnipresent RunData object

A special object, RunData, is passed around through all the methods in the Turbine framework for each browser request. The object have methods that enables developers to access both Turbine and Servlet specific objects and methods. Turbine methods include getters for the AccessControlList, which Action is invoked, getter for an object representing the active user and similar attributes set by the framework. It also includes setters and getters for the selected Screen and Layout. Servlet specific methods include convenience getters and setters for cookies, client (remote) address and so on, in addition to getters for the Servlet request and response objects, central to the Servlet framework.

Jetspeed configures Turbine to instantiate a special RunData class. This enables Jetspeed to augment the methods from Turbine with additional Jetspeed specific methods, for example methods for getting the capability map for the browser.

Actions

When a user clicks a button, link or portlet-control, an *Action* is fired. The parameters sent from the browser includes the action name and which portlet the action concerns. Turbine translates this into a specific action object, and supplies this in the RunData object. This makes for an addressable action scheme, where portlets may address actions to themselves or another portlet.

7.1.3 Framework features

Portlet API

There are two APIs associated with the Jetspeed portal project. The one that is in use in the current development tree (currently version 1.3) is the original one[23], and will be discussed in this subsection. The second is mainly a contribution from IBM, and attempts to make the API detached from the Jetspeed and Turbine frameworks. This API is supposed to be used in the version 2 of the Jetspeed framework, but this version is currently not developed on. The idea behind a decoupled API is to enable other vendors to make a portal server implementing the API without necessarily requiring the Turbine framework. It would also have made portlets portable, enabling them to run on any server implementing the standard portlet API. This new API will be further discussed in the IBM section of the analysis.

In the current API, the interface Portlet must be implemented by all code that should run within the portlet environment. There are several sub interfaces and abstract classes that goes with this base, of special interest are Portlet, PortletControl, PortletController and PortletSet. These relate to each other as shown in Figure 12.

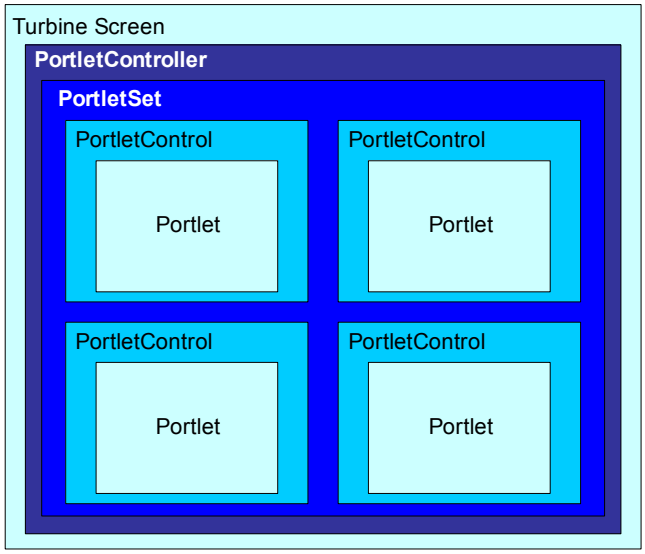


Figure 12 Conceptual relationship between Portlet, PortletControl, PortletSet and PortletController, rendered within a Turbine Screen

The PortletSet is contained within a PortletController. The PortletSet contains all the portlets that should be rendered. The PortletController may be "paned", meaning that it can contain several sets of portlets, selectable via tabs. The PortletControl is

[23] The Portlet API JavaDocs for Jetspeed version 1.3 is available at http://www.bluesunrise.com/jetspeed-docs/javadocs/ (Accessed May 5, 2002)

responsible for displaying the "window" that the portlet's content reside in. Finally, the Portlet is executed and its content is included in the output.

It is worth to notice that the PortletSet and PortletControl are subinterfaces of Portlet. This means that in every position one would expect a portlet, one may instead insert a new PortletSet, and have a set of portlets shown. Thus, as mentioned earlier, one may configure a hierarchy of workbenches within each other. The upper entity in this hierarchy is always a paned PortletController, thus constituting the workbenches of the system. The portlet developers use a special link reference generator to form a reference to the portlet, thus being able to uniquely identify the portlet and send parameters to it from the browser regardless of its position in the hierarchy.

Several abstract classes are defined for each of these interfaces. These classes provide a barebones implementation of the interface, thus making the programming of a portlet easier, as most methods already are implemented in the abstract class.

This API is tied very closely to three frameworks: Turbine, Jetspeed and Element Construction Set, ECS[24]. The ECS toolkit is a set of code that enables a developer to code programmatically a HTML document by making and assembling objects referencing HTML tags like Body, Header and Center in a structured fashion. The link between the portlet API and ECS is that the main content generating method in the Portlet interface, getContent(), is expected to return an ECS element. This dependency is largely deprecated, and is supposed to be taken away shortly [Apache A].

However, the idea behind getContent() is simple, yet powerful. This enables the portlet developer to generate the content in any way he feels fit. For example, the developer may choose to use a template language to render the content. Or he may choose to send the parameters further to a second server for rendering of the content there. This opens for the use of an entirely different programming languages on a different platform, for example PHP, ASP or similar technologies. This is along the same lines of thought that Plumtree uses in their portal system, confer the Plumtree analysis in a later section.

The portlets can get hold of its *PortletConfig* object, providing runtime configuration parameters. This enables, as mentioned, the use of the same portlet code in multiple portlets. For example, a generic portlet for fetching news sources may be configured with the URL where the headlines are found, and which "news-exchange-protocol" this site is using.

XML Configurations and Reuse of Portlet Code

The portlet developer must define and configure the portlets, controls and controllers in the *Registry*. The registry is a set of XML files ending with ".xreg". A *portlet-entry* contains the following elements:

- Classname, the java file that ultimately implements the Porlet interface.
- Parameters, which are accessible through the PortletConfig object.
- Meta info, containing description for the porlet

[24] The ECS's homepage: http://jakarta.apache.org/ecs/

- Role, specifying which role the user must have to be able to view the portlet
- Media, a comma separated list of markups that this portlet can display
- Hidden, whether the portlet should be made available to users

In addition, there is a "type" parameter. The value of this parameter is either "instance", "ref" or "abstract". An instance portlet is a directly defined portlet, and should include all parameters necessary to instantiate a portlet from this configuration alone. A "ref" portlet refers to a parent portlet, overriding any parameters that are both set in the parent portlet and this definition of the portlet. The parent portlet may also be a "ref" portlet, thus enabling a chain of references, ultimately ending in an instance or abstract portlet. An abstract portlet is in effect the same as an instance portlet, but this type cannot be instantiated by this configuration alone. The only way to instantiate an abstract portlet is to refer to it via a "ref" portlet, thus filling in the parameters necessary to configure the portlet.

This is in many aspects an object oriented approach to the definition of a portlet. The "ref" type of portlet makes the defined portlet inherit the properties of its parents, thus resembling the inheritance property of object-oriented languages. In addition, reuse of code is also facilitated by the fact that one java class file may be used in several instance or abstract portlets.

Included with the system are a set of standard portlets. These portlets are supposed to be referenced by a "ref" portlet definition, supplying the necessary parameters so that the portlet displays the correct URL, template or news source. A few examples: FileServerPortlet simply includes a static, configured URL. XSLPortlet fetches a XML file and does a XSLT transform on it with the configured XML styles sheet. RSSPortlet fetches a Rich Site Summary (RSS) file and formats the content to the desired markup, either HTML or WML. RSS is a format that enables for example news sites to export their current headlines. There is also a similar portlet for the Open Content Syndication (OCS) format. VelocityPortlet renders a Velocity template and displays the result. JspPortlet executes a JSP file, and displays the result.

PSML

In addition to the *Registry Markup,* the user's preferences, the so-called *Site Markup*, are the two markup languages that makes up the *Portal Structure Markup Language*, *PSML*[25]. The site markup is stored in separate XML files for each user. These files includes information about which portlets are displayed on a given user's workbenches, which layout is used, what states the portlets are in, and what skin is in use. This is stored in a fashion similar to the hierarchy of portlets and panes that the user has configured.

Services

Services are locateable pieces of code that is configured in the Turbine configuration file. They are accessed in the code using the RunData object that is passed around to every method. Querying this object for a service name returns the associated service if configured.

[25] Portal Structure Markup Language, PSML: http://jakarta.apache.org/jetspeed/site/psml.html

The template based portlets described earlier is using a *template-locator service* to find and load the configured templates, and a *template rendering service* specific to the template language to execute them. A *caching service* provides functionality to the portlet developer to quickly develop programmatic caching behavior for the portlet. There is also a rudimentary *persistence service*, callable from portlets. The portlets may store strings in a map keyed by strings. By invoking the store method on this service, the portlet may at a later login retrieve the same strings.

7.1.4 Software Quality Model References

Fault tolerance

The use of the getContent() portlet invocation principle enables the framework to confine errors. If the portlet that is rendering its content crashes while preparing it, the framework can instead abandon its content and display an error message. However, whether the Apache Jetspeed system actually does this is not known.

Analysability

The software is not very documented, and the portlet API and both of the underlying frameworks are considerably intertwined. This makes for tricky bug hunting, as there is no clear distinction of where error sources may be located.

Changeability

The software is changeable in every way since it is an open source product. The entire product can be modified to the needs of the situation. Used as is, the framework is also very adaptable, with many possible configurations.

Stability

Apache Jetspeed is in beta revisions as of this date, and is not considered ready for production usage. In addition, as low-level features as the security model are still being revised and discussed[26]. This is not very promising regarding the stability of the code.

Adaptability

The Apache Portal supports multiple markup languages, specifically is WML for WAP devices supported by default. Each portlet must specify which markup languages it support, and then format its resulting output according to the connected device.

Conformance

The Apache Turbine and Apache Jetspeed projects need a servlet container to run on, and it does not need the full J2EE environment. The portal does not require any specific vendor's servlet container, and is developed to be standards compliant. It is currently developed using the Tomcat 4 engine, which is the reference implementation of the Servlet 2.3 standard, but can run on any Servlet 2.2 standard compliant server too.

[26] For Example: New security proposal, May 22, 2002: http://nagoya.apache.org/eyebrowse/ReadMsg?listName=jetspeed-dev@jakarta.apache.org&msgNo=4273

7.2 BEA – WebLogic Portal

BEA is one of the best-known Java application server providers. BEA states on their web site[27], "BEA is the world's leading application infrastructure software company with more than 12,500 customers around the world, including the majority of the Fortune Global 500." In the Metrics of Success section, they boast that they have, according to every major industry analyst, the world's number one Java application server. They have more than 2,100 partners, which offer more than 1,000 "Built on BEA™" applications.

BEA's family of products is called WebLogic. The BEA WebLogic Enterprise Platform embraces their full set of products. This platform includes WebLogic Server, WebLogic Portal, WebLogic Integration and WebLogic Workshop. BEA states in the "WebLogic Portal Datasheet" [BEA A], "BEA WebLogic Portal is a cornerstone of the BEA WebLogic E-business Platform". The WebLogic Portal replaces BEA's former products WebLogic Campaign Manager and WebLogic Commerce Server. These facts have many implications for the focus of features in the portal framework. In this analysis, only the WebLogic Portal features will be investigated, and the e-business features will only be skimmed.

BEA's portal solution, including the application server and a truly extensive documentation pack (~45MB zipped), is offered as a 30-day free trial download at their web site. This proved invaluable, as it gave good insights in how the portal operates in conjunction with the other components of the system.

Figure 13 Screenshot from the demo portal supplied with the trial version (from test installation)

[27] About BEA: http://www.bea.com/about/index.shtml

7.2.1 Architecture and Architectural Features

Java 2 Enterprise Edition foundation

The WebLogic Portal is built on Java technology, and is designed for the Java 2 Enterprise Edition. Nevertheless, version 4.0 of the portal server requires the BEA WebLogic Server 6.1 with Service Pack 1. In fact, BEA points out in the architectural overview that the portal is purely an extension of the WebLogic Server.

Where J2EE is designed for compatibility between the different vendors' servers, BEA has instead made their portal framework dependent on their own architecture. This way of thinking is in stark contrast to other, independent portal server vendors, for example Epicentric, which focuses on making their portal server capable of running on multiple vendors' application server. BEA is in every aspect a typical representative for the "Existing vendors, new market opportunities" categorization explained in Chapter 5.4 – "Portal Categorizations".

BEA does not have any database server of their own, and the selection of databases on which the system runs is therefore much wider. Supported databases are IBM's DB2, Oracle, Microsoft's SQL Server and Sybase. In addition, Cloudscape, a pure Java RDBMS, is shipped with the product. This is primarily to be able to run the system without any other databases installed.

Overview

The portal system provides a set of services. These are tuned towards "… efficiently build, launch and maintain high-performance e-business sites." [BEA B]. There are four main services:

- *Portal services* provide means for creating, deploying and managing multiple enterprise portals, targeted to the corporation's different user bases.
- *Personalization services* make the portal user experience tailored to each user's role and needs. It delivers dynamic and personalized web content.
- *Campaign services* are provided to target advertising, e-mail and product discounts to specific audiences.
- *Commerce services* are building blocks and functions for e-commerce, for example shopping carts, product catalogs, transactions and order fulfillment.

Of these, only the portal services are of interest, while the other three are not integral to the portal architecture and its concepts and features. They will be briefly mentioned throughout the analysis.

The WebLogic family of products, including the portal, is extensively using the paradigms and features of the Java 2 Enterprise Edition framework. Concepts from this framework, for example "web application", "enterprise application", "enterprise java beans" and "deployment descriptors", is used frequently. The entire portal is made up of standards-compliant elements, to such a degree that one wonders why it is required to run on the WebLogic server. Appendix B contains a rudimentary overview over the J2EE platform.

BEA use a graphic user interface called E-Business Control Center, EBCC, to create and edit many of their servers' properties and the functions within the framework. The primary function of this console is to create and edit XML files within the system. It can also, using a special synchronization mechanism, communicate directly with the WebLogic Server and thus update applications on the fly.

Webflows

To understand the low-level architecture of BEA's portal, one important feature of their system must be briefly described, namely Webflows. This feature helps developers make extensive multi-screen applications. A webflow is based on a flowchart-style model. Figure 14 shows a part of an extensive webflow.

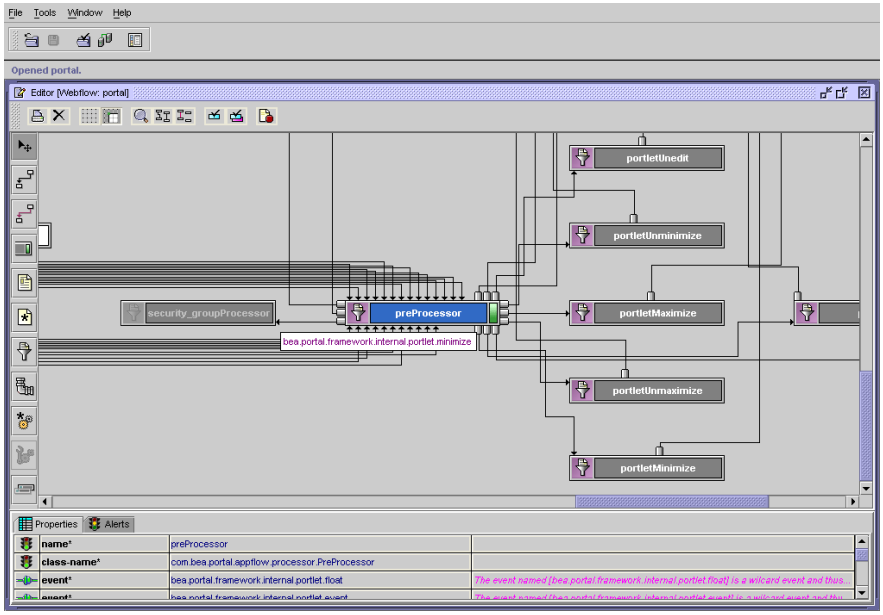


Figure 14 The BEA E-Business Control Center (EBCC) showing a part of the portal Webflow

The most important entities of a webflow are Presentation nodes, Processor nodes and Events. A presentation is a user view, and is stored as a JSP file. A processor is some Java class file that does business logic, retrieving and storing data and taking decisions. Events are fired from both presentations and processors. An example follows. A customer has some decision to take, and his browser is showing him a screen with two buttons. This view is presentationX, which happens to be the start point of a webflow. The customer clicks the button1, thus invoking the "button1.clicked" event. This event is tied, by the webflow, to processorA, thereby transferring control to this piece of Java code. If the customer clicked button2, the "button2.clicked" event would be fired, thus giving some other processor the control. ProcessorA might decide, based on some values from a database, whether it should fire event "customer.WellOff" or "customer.BadFinances". These events are tied to presentation2A or presentation2B respectively, giving the customer a new view.

Using these features, the developer can construct large and complex web applications, freeing him from the burden of coding the transitional logic. The developer may instead use the graphical tool in the E-Business Control Center to design the flow. The flow is stored as an XML file, and loaded by the WebLogic Server's flow control

service. The processors and presentations may, if well coded, be re-used in other settings, tying the events to different presentations or processors.

Architecture of the portal

The portal is in effect simply one big webflow. An overview over the entire flow is shown in Figure 15.

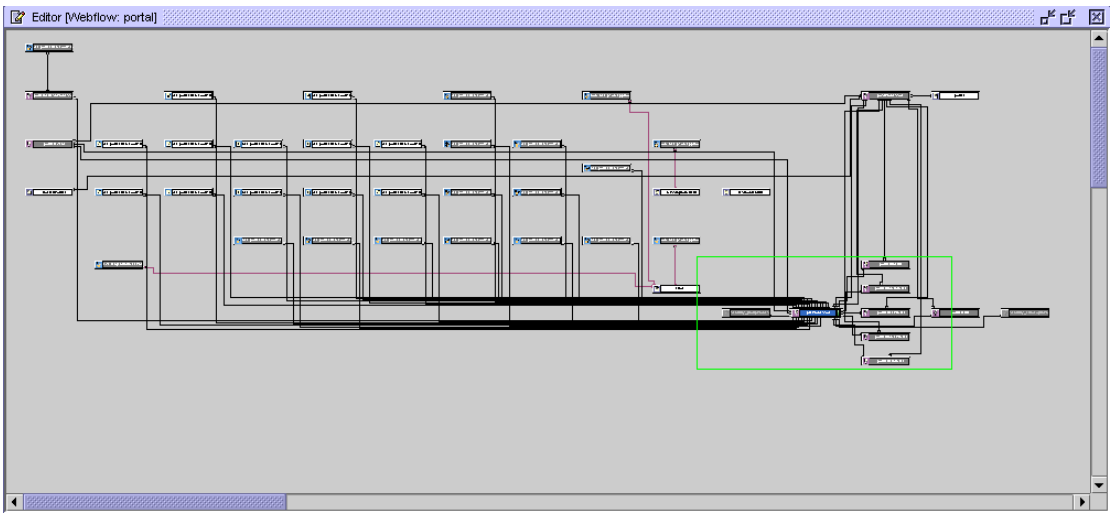


Figure 15 Overview over the portal Webflow. Green rectangle is shown in Figure 14

The BEA portal consists of a set of presentation JSP files and set of processors, which are tied together by a set of events. In addition, there is a system-internal Java package "com.bea.portal" that contains the state handling framework of the portal. Some EJBs are also in action.

The major part of the portal logic lies in the central "preProcessor" and its helpers processors shown in Figure 14. All these processors end up in "postProcessor", which in turn send the user to the "portal" presentation node. This last node consists of portal.jsp, the central portal view renderer. This file, in turn, includes files that make up the portal page, including banner, title bars, and the content.

The content of the portal page is decided by the state of the portal, which is kept with the portal framework. This is either one of a set of customization pages, where the user chooses things like the layout of the portlets and which portal pages he wishes to have present, or it is one of a set of layout files.

The layout file contains placeholders for the different columns of portlets. These layout files may be "two-column", "three-column" and so on. Developers may also make their own layout files, including placeholders to let the portal system place the portlets. The names and meta-data about the portlets that should be placed in each column are fetched from the database. The file portlet.jsp is responsible for rendering the portlets, unless the portlet is in its "floating" state, in which case the

floated_portlet.jsp takes care of the rendering. The portlet.jsp file is invoked repeatedly, with different arguments, once for each portlet in each column.

Every aspect of the portal presentation and functionality may be edited and tailored to the needs of the corporation. Since the whole portal is merely made up of a set of JSP files, graphics and a large webflow, a developer may even choose to implement the portal from scratch. Editing the portal webflow is not recommended, as this webflow is of a complicated nature [BEA C]. However, the developer guides does indeed state that if the developers are "well versed with webflows", they can even customize this part of the portal.

Architecture of the portlets

A portlet is a set of JSP files associated with some XML meta-data. The main parts of one portlet is shown in Figure 16.

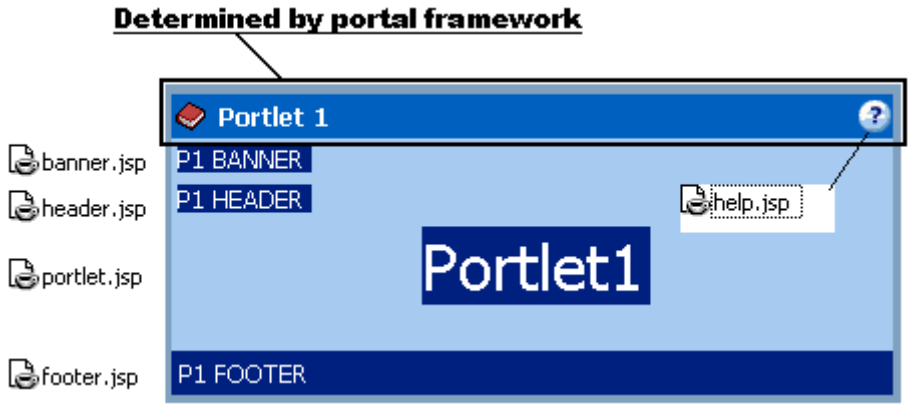


Figure 16 Main parts of a portlet, filenames are local to the "portlet1" portlet. By clicking on the help-icon, the help.jsp file is run and displayed in a disconnected browser window. (from [BEA B])

These elements and several other attributes comprise a portlet definition. Other elements include parameters like "editable" (the user may edit the portlet, the parameter is which JSP file should be shown in edit mode), "floatable" (the user may disconnect the portlet, yet another JSP file), and attributes like "minimizable" (Only the titlebar is showing) and "maximizable" (using the entire work area), and whether login is required. The portlet.jsp mentioned in the last subsection is responsible for invoking these various JSPs and taking into consideration the numerous attributes.

Portals as Web Applications

When installing the WebLogic Portal, several enterprise applications are installed as well. One of these is the "portal" enterprise application. This includes a large set of enterprise java beans and three web applications. Of these, the "stockportal" web application is a default portal, including everything needed to make a portal. This layout is shown in Figure 17.

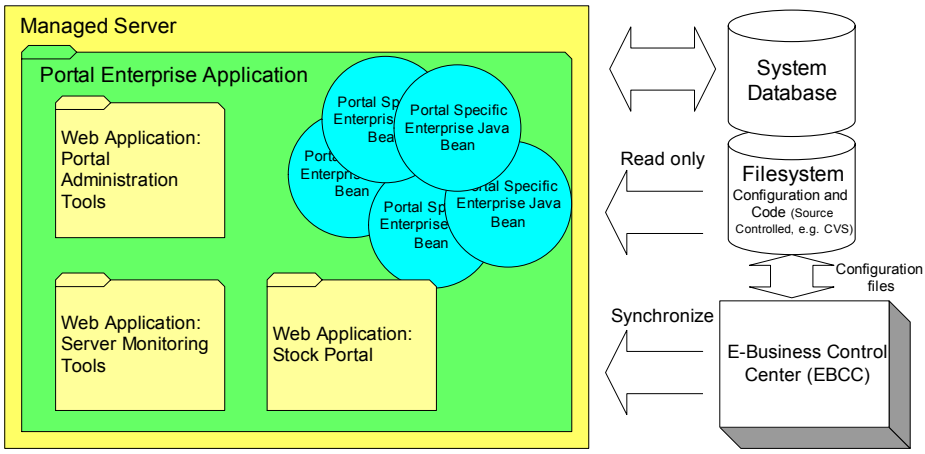


Figure 17 Relationships between the different entities in BEA's portal system

The documentation [BEA C] states that to make a new portal, a copy of the stock portal directory is made, and placed in new directory on the same level. The name of the directory is the name that the portal web application gets. The WebLogic server is made aware of the new portal web application within the portal enterprise application by configuring it using the E-Business Control Center. This is shown in Figure 18. One server installation may thus be used to deploy multiple portals.

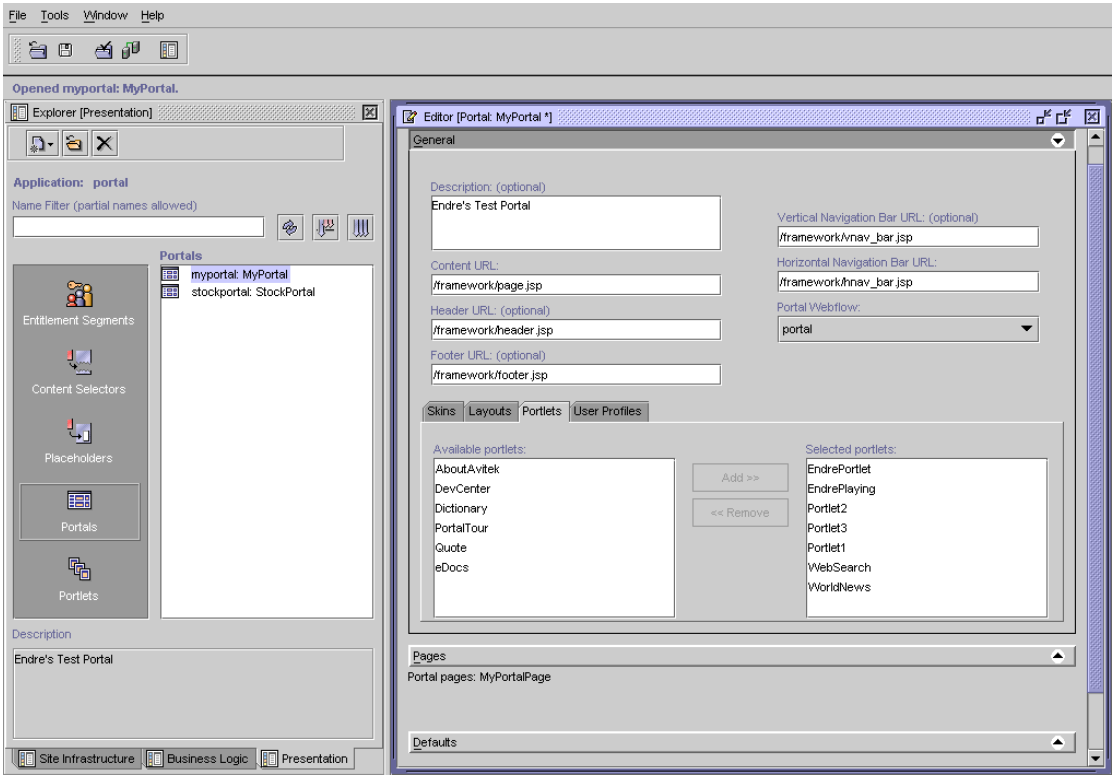


Figure 18 The new MyPortal portal web application, with the configuration screen to the right.

Different resources are scoped within different sections of the J2EE hierarchy, confer Appendix B. Portal web applications are scoped within an enterprise application. Group portals are logical divisions of portal pages, thus scoped within a portal web application. This is shown in Figure 19.

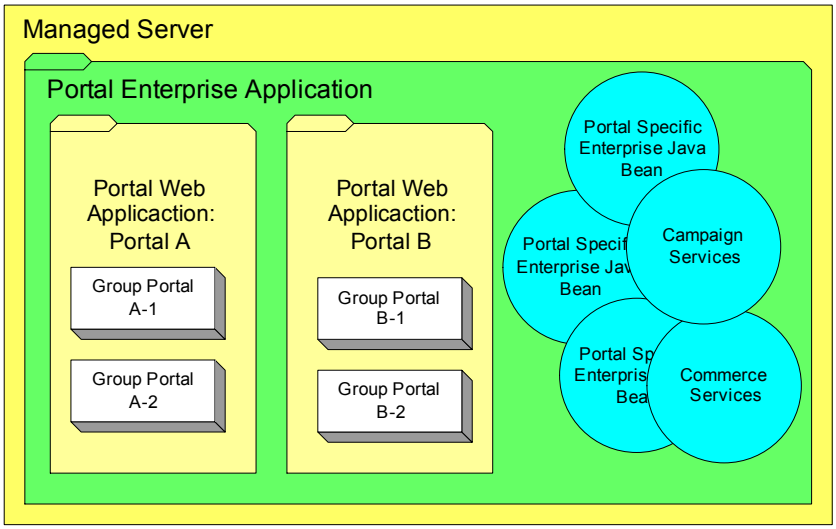


Figure 19 Scope of Group Portals vs. Portal Web Application vs. Enterprise Application

Webflows, portlets, skins and layouts may be shared between the different group portals. They cannot be shared between different web portals, except by copying the physical files consisting the resource from one web portal application's directory to another. These files includes the resources making up the portlets, e.g. JSPs, Java class files, graphics and HTML. In addition, the XML files, making up resources as webflows and portlet configuration, must be copied or regenerated using the EBCC.

However, the Campaign and Commerce services are scoped within the enterprise application, existing as enterprise java beans. These are thus shared between all the web applications.

7.2.2 Framework concepts

Portal Administration

The portal administration consists of managing the relationship between entities such as users, user groups, portlets, skins and layouts.

The WebLogic Portal recognizes three types of administrators. These are *System Administrator* (SA), *Portal Administrator* (PA) and *Group Administrator* (GA).

System administrators, which are members of the special SystemAdministrator group, have access to manage all resources, and thus all portal web applications, within an enterprise application. He may use a special web application within the portal enterprise application, called WebLogic Portal Administration Tools ("portalTools"), to control all aspects of the different portal web applications. This application's start page is shown in Figure 20.

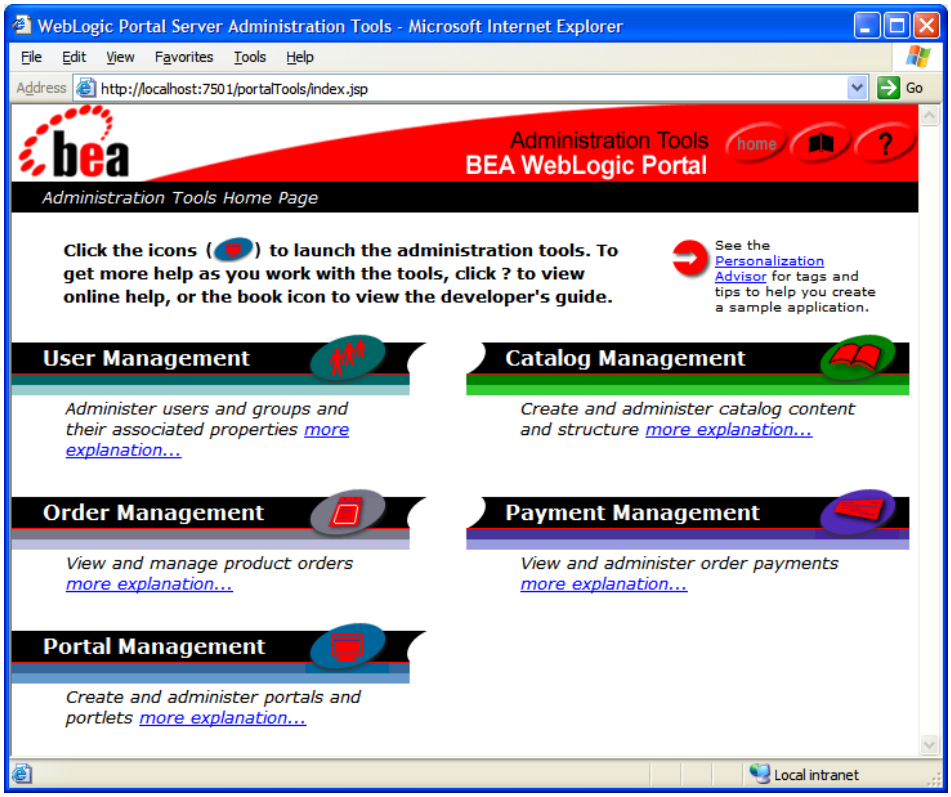


Figure 20 The WebLogic Portal Server Administration Tools

The system administrators are the only ones that may make new users and groups. They may also deploy new portals. A system administrator may create new portal administrators and group administrators. He may further choose which tasks these new administrators can do, and whether they are allowed to further delegate these tasks. A portal administrator may administer the properties and make new administrators for the portals which he have access to, but only if he have gotten the necessary rights from the administrator that created him. Users must belong to a special group called AdminEligible to be granted any administrator privileges. The WebLogic Portal Management start page is shown in Figure 21.

A group administrator may create new group administrators, and control the customization for the group portals he is assigned to, again only if the administrator that created him gave him the necessary rights. The customization includes which portal pages, portlets, layouts and skins that are visible, available and mandatory. Visible means that the portal page or portlet is default present, available means that the user may customize it onto his workbench, while mandatory speaks for itself. He may also choose the default settings for all these parameters. This is called group customization.

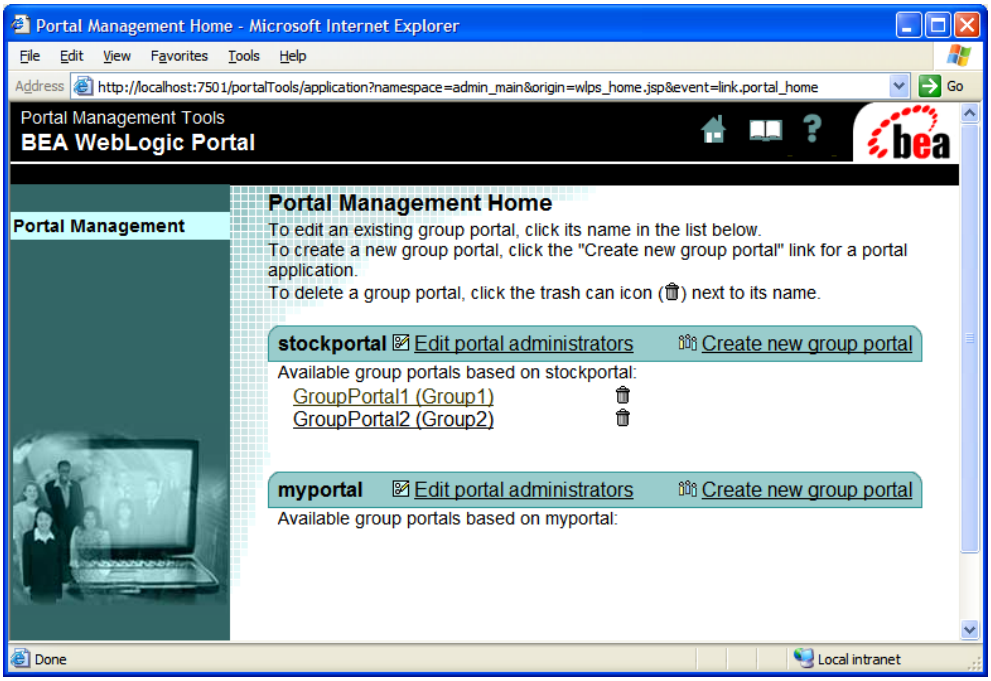


Figure 21 The Portal Management Tools – viewed by the System Administrator, thus showing all the portals in the enterprise application.

The delegation of rights is done using the "Delegate Administration" page, shown in Figure 22.

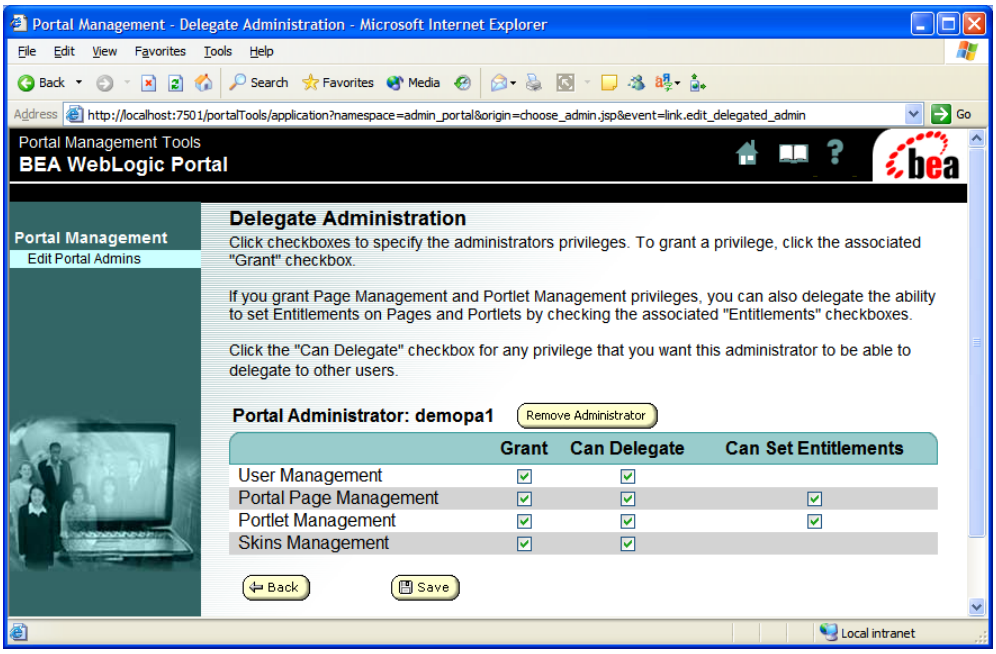


Figure 22 The Delegate Administration page.

Customization

Customization refers to the process of letting an administrator for a group, or the user himself, configure and manage the looks, layout and pages visible in the portal. It also includes any portlet specific configurations, where the portlet's view is changed according to a user or group's preferences.

A feature that definitely lacks from BEA's portal system is the ability for a user and group manager to make new portal pages. Most other portal systems have a concept of "MyPages" and "GroupPages". In the WebLogic Portal, the users and group administrators may only choose between pages already configured into the portal by the portal system administrator.

Personalization

Personalization is similar to the customization in that it changes the view of the portal for a user or a group of users. However, where customization is an explicit process, where a user or a group's administrator chooses what to see, personalization is a more implicit process. The view of a portal, including which skins, portal pages and portlets that is in use, and which content the portlets are showing, is changed according to properties of the logged in user. A customer gets to se different pages, portlets and content than a partner do, and a good customer will get special offers which other customer do not get. This is done using the personalization package [BEA D]. This framework may be licensed separately, but is included with the portal license. Both attributes and entitlements control the visibility and availability of portal pages and portlets. However, where attributes only chooses whether a portal page or portlet is visible, available or mandatory within one group portal, the entitlements give much finer control.

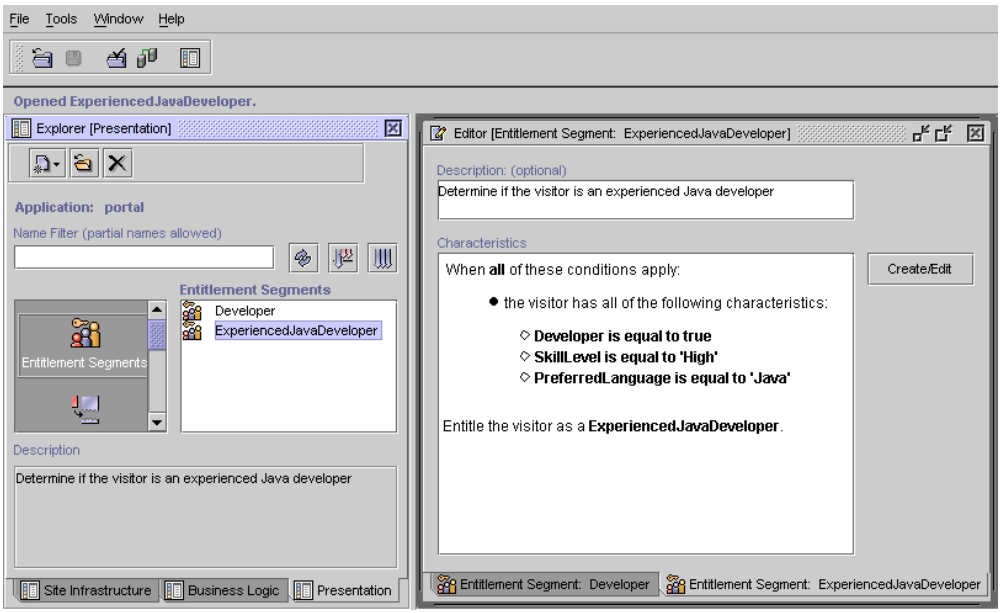


Figure 23 Viewing entitlements within the EBCC

Entitlements are based on the user's profile, date, time, properties of the user's session and HTTP request. The EBCC view in Figure 23 shows the entitlement "ExperiencedJavaDeveloper", and which conditions which must be met to be entitled with this description. The entitlement editor is shown in Figure 24, and shows all the different conditions that may be imposed on an entitlement.

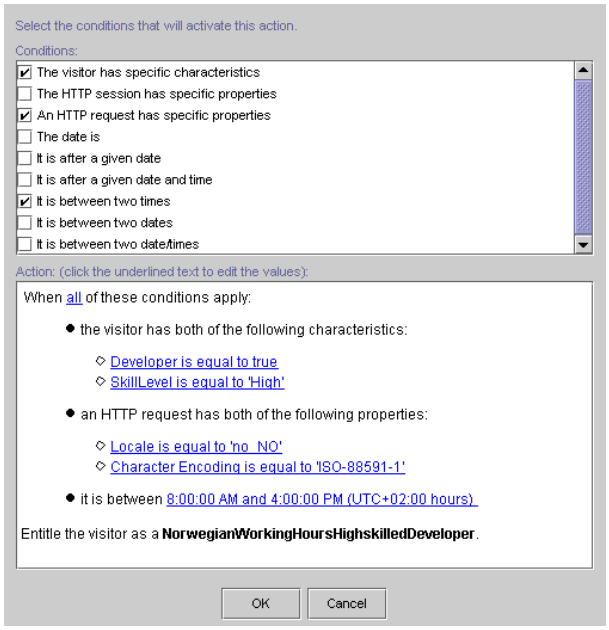


Figure 24 The entitlement editor

Using the portal's management editor, the administrator can choose whether a portal page or a portlet can be viewed and whether it can be removed from the user's view, by either granting, denying or abstaining from altering the value of the right, based on the user's entitlements. The content of the portlets may also be changed based on entitlements, this is gone into further detail in the "Framework features" subsection.

7.2.3 Framework features

JSP Tag Libraries, Internationalization

The programming of portlets is done using multiple JSP files per portlet. The JSP files may contain JSP tags, accessed by importing tag libraries into the JSP file. Confer Appendix B.

The WebLogic portal includes a set of tag libraries to enable the integration of the portal framework's features with the portlet logic. WebLogic supplies dozens of tag libraries to make the development of JSPs easier. For example, there are tags to refer to the user's session, to the request and the response objects and similar JSP and Servlets related objects. There are several libraries to query the personalization system mentioned in the last subsection, including content management, ad placeholders, and user management. These tags may change the content in a portlet based on the different properties of the user and which entitlements that are in effect. Tracking libraries enable the developer to monitor user behavior and user events, like clicking on a product or content. There are also tags that enable the developer to efficiently manage a product catalog; this goes in under the commerce services mentioned in the beginning. Two simple tags involves internationalization; <i18n:localize> tag, used to define the language, country, variant, and base bundle name to be used throughout a page when accessing resource bundles via the <i18n:getmessage> tag.

In addition, three of these libraries are special to the portlets environment. The Portlet library includes tags for referring to portlet webflows, generating HTML forms and validating the input from the user and for referring to the portlet related events like

maximize and unmaximize. The Portal library similarly has tags that refer to the portal's webflow and to generate portal-scoped forms. There is also a tiny utility library used within the portal's own framework.

Webflows within the portlets

The webflow system is not limited to full-page views, but is also able to keep a flow of operation within single portlets. This system also enables multiple portlets to communicate to some degree. This is facilitated by the fact that if one portlet gets, for example, a minimize event, all other portlets on the page would get a *refresh event*, which are accompanied by the portlet namespace and the origin of the event.

7.2.4 Software Quality Model References

Fault tolerance

The total reliance on JSP makes for tricky coding, and unversed coders may make portlets that may crash under given circumstances. As the JSP files are included recursively and sequentially, each unit must be totally correct in execution. If one unit crash halfway into its markup generation, the resulting page might very well contain inconsistencies. This can lead to the situation where the entire portal page is not able to render on the client's browser until the error is cleared.

Changeability

The portal system is a J2EE enterprise application, and each part of the system is defined using the standard J2EE deployment descriptors, in addition to the BEA specific deployment descriptors. The JSP files controlling the rendering and the webflow describing and controlling the flow of operations are also readily available. Thus, the system is changeable in most every way.

Analysability / Testability

The BEA J2EE server has some interesting statistics associated with all running servlets and JSPs. For each of these, an access count is kept, and the low, high and mean processing times are updated for each access. This is an extremely valuable feature when a developer is trying to optimize a page, or trying to make out why some workbench is too slow.

However, the multiple recursive inclusion of great many JSP files makes the system hard to decipher. If, for example, a mismatched element is discovered in the resulting workbench markup language, it is often virtually impossible to decide where the element comes from.

Adaptability

BEA has some support for internationalization, however, there are only two tags in the tag libraries that tend to all internationalization, and those only involve the language. Date, times and numbers seems to not have any helper functions, and the user profiles does not have any locale-specific properties embedded.

Conformance

The BEA WebLogic portal system specifically requires the BEA WebLogic J2EE server, and thus does not conform to the J2EE standard. However, the BEA

WebLogic J2EE server conforms to the J2EE standards, thus being able to run components conforming to these standards.

Scalability

The BEA WebLogic J2EE server is scalable, and can run on multiple physical servers. The portal system inherits this feature, and it is thus possible to handle very many concurrent users.

7.3 Epicentric – Epicentric Foundation Server

"Epicentric is leading the portal market evolution" is to be read in Epicentric's "What's New" white paper for their Foundation Server 4.0. Another bold statement from the company is their trademark "Epicentric – the Portal Platform Standard".

Epicentric have over 400 installations spread over about 200 customers. Epicentric's sole focus is the portal market. Epicentric have 335 employees, and boasts over 100 partners. Their focus is the Global 2000 companies[28].

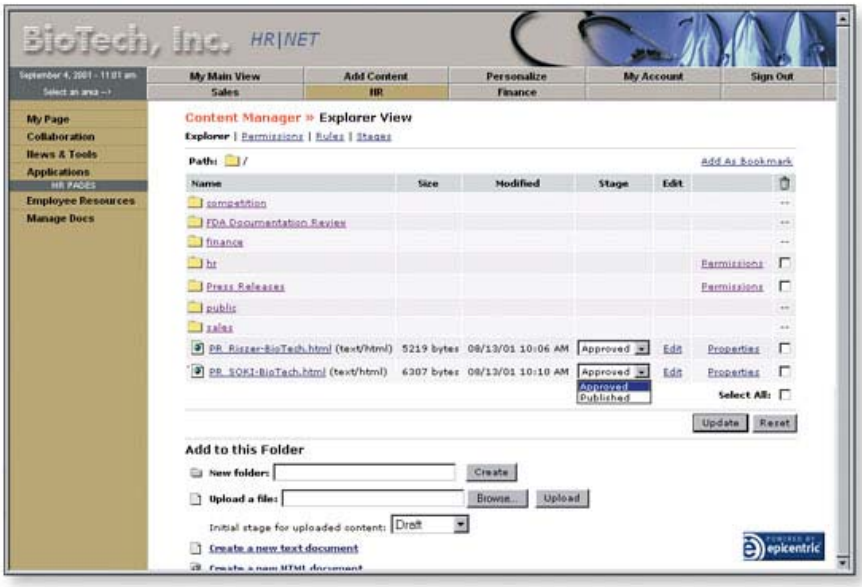


Figure 25 Screenshot from Epicentric Portal, Extended Content Management (from Epicentric webpage)

7.3.1 Architecture and Architectural Features

Java 2 Enterprise Edition

The *Epicentric Foundation Server* is built on Java technology, and runs on several J2EE application servers. Epicentric does not provide their own J2EE server, and it is therefore important for them to make the user able to choose between the different J2EE server vendors. They have tested their system on multiple vendors application servers, databases and LDAP servers. The list includes BEA WebLogic, IBM WebSphere and Macromedia JRun for application server, IBM DB2, Oracle's 8i, Sybase and Microsoft's SQL Server for database, and Netscape and iPlanet's

[28] Epicentric Company Facts http://www.epicentric.com/company/fact_sheet.jsp

Directory Server, Novell eDirectory and Microsoft's Active Directory for the optional LDAP server[29].

By using the J2EE platform, Epicentric instantly gains scalability by using a scalable J2EE server. In addition, some commercial J2EE servers provide for seamless fail-over in case of application server (or JVM) crash. BEA WebLogic™ and IBM WebSphere™ do this by using the attached database to store the session data between requests[30]. This is described thoroughly in their "Scalability and Capacity Planning Overview" paper [Epi A].

Modules and Pages

A *Module* is the term Epicentric uses for a portlet. The modules are arranged on a *Page*. Modules may be programmed in Java using the native server API, or using the Modular Web Services framework, which is described later.

Portal Sites

The Epicentric Foundation Server 4.0 allows several *Portal Sites* to be hosted on one server installation. A site within the server is a collection of portal components, including pages, modules, navigational tools and a theme, and administrative functions given to a designated portal administrator. The sites are hosted on separate domains (e.g. www.divisionA.com and www.divisionB.com) or as separate paths beneath one domain (e.g. www.corporation.com/siteA and www.corporation.com/siteB). The set of portal sites that are hosted on one server is called a *Portal Network*. A user may use multiple sites on the system, depending on their permissions. The portal network also enables server and site administrators to share and re-use site components between sites.

Enterprise Portal Management

From [Epi B], "Epicentric Foundation Server 4.0 is the industry's first Enterprise Portal Management (EPM) system. Beyond simply managing employee portals, Epicentric Foundation Server 4.0 is specifically designed to enable customers to rapidly build, customize and manage multiple portal initiatives across the extended enterprise."

Epicentric argue that large corporations often has deployed several portal installations around the same time at different departments [Epi C]. This has happened because a bottom-up approach for portal implementation and installation has been employed. Each department wanted their own solution, driven by their own needs, and they wanted it quick. Often departments have chosen different platform standards, giving a heterogeneous mix of portal platforms.

Now these corporations are in the same situation that portals were conceived to remedy; multiple intranets spread across the departments, without the possibility to share information. This prevents the true value of a business portal (which is

[29] Tested Platforms Matrix: http://www.epicentric.com/solutions/efs_testedplatforms_40_sp3.jsp

[30] This seriously impact performance, though, as every session must be serialized to a sequence of bytes, stored onto the database persistent store, and then finally read and de-serialized again on every request. The author finds this approach somewhat counter-productive, as it only moves the single-point-of-failure "one step backwards", completely relying on the database. This definitely hints at a fact that most vendors' database solution is more mature than their application servers are.

Epicentric's favored term) to emerge, as the "information-silos" still are unconnected. To reach all information, the users have to roam from site to site.

This is where the "next generation portal solutions" is needed. Epicentric says that to implement an Enterprise Portal Management System, one will need an enterprise wide portal platform standard. This system will have to have integration capabilities that will unify a large set of different portal solutions, such as IBM, Plumtree, SAP, Lots, Oracle and Microsoft Exchange. Since "Epicentric is the only portal vendor to provide the complete enterprise-wide portal platform standard required for an Enterprise Portal Management System." [Epi C], the solution is self-apparent. This white paper does not explicitly tell how this integration of portal deployments is to be done. It still seems like the corporation is required to leave the earlier portal deployments behind, and migrate them all to one installation of Epicentric's "unified portal system".

Multi-Level Delegated Administration

A server administrator may create a set of portal sites for different business groups, for example different departments or different entities in the extended enterprise. The administrator can further grant administration rights to the different sites to the sites' respective owners. The server administrator may restrict the site administrators' administrative capabilities using *Permissions*. The site administrator may also elect to share the responsibility between multiple co-administrators by restricting administrative rights within the portal site. This is called *Collaborative Site Management*. Each portal site is managed from its dedicated *Site Console*. The console lets the site administrators manage all aspects for a portal, including the workbenches, portlets and styles. All portal sites may also be monitored or managed centrally with the *Server Console*, which is located within the *Portal Command Center*.

Content Management, eXtended Content Management

XCM is an integration layer for already existing document management systems. Epicentric reasons that many corporations already have one or several document management systems in place already. XCM integrates multiple systems, by aggregating the content and presenting it to the user through the portal's user environment.

XCM is delivered as a suite of modules and core system services. It provides a document repository and a system of connectors to other document management systems. Epicentric provides XCM connectors to "solutions like Documentum and Interwoven" [Epi B]. It can also access files from file systems. XCM is described as a "Content Management Middleware" layer. It provides methods for creating new or specialized display Modules. Back-end system connectors may be made to integrate to proprietary or legacy document systems.

XCM handles both content and documents. It builds upon existing functionality in the server to basic workflows, staging, approval and expiration of content.

Search and Taxonomy

The taxonomy and searching capabilities of Epicentric's Foundation Server comes as a module. The system works in concert with XCM to organize and categorize

information assets and Web content in a secure repository. The different elements of the taxonomy system are protected by the permissions systems, and thus enable the administrator to delegate the creation and maintenance of its structure and content to selected users.

The taxonomy's directory-like structure is populated with *Cards*. These contain links and meta-data that refer to documents on web servers and objects stored within the XCM structure. The user may browse this structure as he would browse a file structure, and access the objects and their metadata directly.

The searching facility is built upon the Inktomi Enterprise Search Engine. It integrates with the taxonomy, enabling user to search for objects based on their meta-data and content. Support for other search engines will be offered in the future.

Statistics and Business Metrics

Epicentric's term "Business Metrics" is referring to statistics specifically tailored to measure the effectiveness of different portal sites. The framework gives the administrators the possibility to develop their own metrics, both for the custom modules and for the framework itself.

Included in the Epicentric Foundation Server is several metrics, both static and periodic. The static measures include number of registered users, pages and sites in the system. The periodic measures include number of users using each module, active users, total page views, front page views and maximum and minimum concurrent users. These latter figures are all measured within some specified period.

7.3.2 Framework concepts and features

Portal Site Creation

To help create and assemble new portal sites, Epicentric provides several tools within their system:

- *Site Creation Wizard* walks the administrator through the steps required to deploy a new site. These steps are:
    1. Site info: choose the name and the URL of the portal site
    2. Pages: choose which Workbenches that should be available
    3. Modules: add portlets to the selected workbenches
    4. Appearance: apply a theme to the portal site
    5. Site finished, preview.
    6. Deploy.

- *Shared components* refer to portal components that may be shared and re-used between multiple portal sites that reside on one portal server. Shared components include modules, pages, themes and other user interface elements.

- *Rapid Site Assembly* allows administrators to build portals from repositories of pre-built, reusable components. New portal sites may be developed and deployed "within minutes instead of months".

- *WYSIWYG creation of menus and submenus* enables the site administrator to create or edit the menu structure that users of the portal use to navigate through the site. The menu items may reference a wide range of items, including links to the different workbenches, existing enterprise web pages and internet web pages.

- *Unified Site Themes* are the portal sites' look-and-feel. Visual elements of the *Site Theme* include portlet appearances, *footer*, *header*, *site controls* and *navigation blocks*. There is also a concept of a portal site's *grid*, which is the physical layout of these other elements.

Export/Import of Portal Site Configuration and Portal Components

It is possible to export and import the configuration of entire portal sites, using XML files as the configuration transport medium. In addition, also site components can be exported and imported. The server console is used to access these features. These functions makes backing up the portal's definition and configuration easy, but also facilitates the development and staging of portals sites, including portal sites that already are in production.

Staging is for example done by copying the live configuration to a new portal site, possibly residing on another server. This new site may be reconfigured and tested, and then loaded into a staging server for final user testing. Finally, when the user groups are satisfied, the configuration may be loaded back onto the production system.

Modular Web Services and Web Services User Interface

Using the web services protocols, Epicentric's *Modular Web Services* technology, MWS, enables Modules to be hosted remotely and serve views to the portal site. This technology may be viewed as extending Epicentric's Module API to a remote host. The SOAP/XML based protocols distances the implementation of the module from the server platform, enabling MWS to be written in other languages than Java. Epicentric have made MWS versions for both JSP and ASP.

Web Services User Interface, WSUI, is an exciting standard initiative proposed and driven by Epicentric. It is developed because of the fact that Web Services as they stand today is made exclusively for computer-to-computer communication. The protocol would be excellent for remote portlets, though, and this fact is what has made Epicentric develop this standard. The idea is that the WSUI layer defines the user interface and the methods that should be run on the remote Web Services interface when the user makes any actions on the client, for example by clicking on a button, or submitting a form.

This initiative has gotten industry attention, and several of Epicentric's customers and partners have teamed up behind it and established a working group. There is a dedicated website for the work at http://www.wsui.org/. However, the standards body Oasis[31] has decided to initiate the exact same type of protocol, this called Web Services Remote Protocol, [WSRP]. Epicentric have joined this working group, submitting their work as a contribution to the new standard.

[31] "Oasis is an international, not-for-profit consortium that designs and develops industry standard specifications for interoperability based on XML". Website: http://www.oasis-open.org/

Visual creation of modules – Epicentric Foundation Builder

This is a module that help "business users" make Epicentric modules [Epi D]. It has several tools that make the development and generation of modules a more visual experience. Several wizards and other browser based tools leads the user through the design of the application. Reports are made with the help from a query builder. The users of the application can later filter and sort data within their personalized page. Forms are built with functions like 'Add Button" and "Modify Input Field", and data verification rules are built and applied to the form. Remote resources may be leveraged using web services technology. The application is generated, and can be deployed into one or several of the portal sites residing on the server.

How complex applications the application builder can generate is not mentioned, and it is most likely confined to relatively low-complexity "forms-based" applications with data entry and modification, and querying of existing data.

Internationalization and Localization

The Epicentric Foundation Server is internationalized. Portal sites may be deployed in any language. The administrator defines which languages are available for the entire server in the Portal Command Center. The language a user sees is decided based on the locale settings in their user profile. The user may choose between the languages enabled by the administrator.

The components of the portal may be internationalized interactively using the Site Console. In addition, administrators may bulk update every translatable component of the portal by uploading a resource bundle containing all the language strings for one component for one language. Export facilities for the resource bundles are present, thus enabling a template resource bundle to be exported for translation outside of the portal system. Localizable components include modules, pages, styles and user interface components. The localizable components may be protected by permissions, giving the appropriate group of people access to localize components to their respective language only.

Epicentric focuses a lot on the language aspect, but mentions that the server in addition is able to handle dates and times, and international character sets.

Caching API

The Foundation Server has core support for caching of components. The idea behind caching is that the content of some components may be non-trivial to render, requiring significant amounts of processor time and input/output activity to be produced. The caching system is exposed as a *Caching API*, giving the module developers the choice of how caching should be performed.

The developer may choose expiration times and behavior for cached objects. The server has a concept of *Passivation*, giving the developer a choice of where objects and content should be stored when not needed by any user. By default, this is to the server's back-end database, but the developer may choose to use a specified storage system, or not use the passivation mechanism by just keeping the object in memory. The objects may be cached by any combination of keys, for example user-ID, module and/or domain.

7.3.3 Software Quality Model References

Adaptability

Epicentric mentions both character sets, dates, times in addition to languages as localization possibilities.

Conformance

The system can run on several vendors' J2EE servers, thus conforming to the J2EE standard.

Scalability

The Epicentric system runs on several vendors' J2EE servers, of which several are scalable. The Epicentric system inherits these features, and is thus scalable.

7.4 IBM – WebSphere Portal

International Business Machines, IBM, is a computer industry giant with a history going back to before 1900. The company was officially founded as the Computing-Tabulating-Recording Company (CTR) in 1911[32]. At their website, they state, "At IBM, we strive to lead in the creation, development and manufacture of the industry's most advanced information technologies, including computer systems, software, networking systems, storage devices and microelectronics." IBM's 2001 numbers are enormous; 319,876 employees and a revenue of $85.9 billion, of which $7.7 billion were net income[33].

IBM is heavily involved in many areas of computer technology, something that their "Products and Services" pages clearly indicate[34]. IBM is committed to the Java technology, as well as several Open Source initiatives, including Linux[35]. Several of IBM's technologies are for example Linux-enabled, including the Java based WebSphere platform.

IBM's WebSphere family of products includes a range of application servers, several of which are J2EE compliant[36]. Amongst WebSphere's components is the IBM WebSphere Portal[37]. At the time of writing, the version 2.1 of the WebSphere Portal Family is no longer being marketed, and being replaced by the WebSphere Portal for Multiplatforms V4.1. This version is not available until May/June, thus complicating the review somewhat. However, technical information is already available at the portal website.

One of these documents is the "Portlet Development Guide" [IBM A]. This guide in detail lays out the portlet API implemented in the portal server. This portlet API has been influenced by discussions being held on the Apache Jetspeed mailing lists, mainly from October 2000 to around March 2001.

[Screenshot is not available, as the new version was not released at time of writing.]

[32] History of IBM: http://www.ibm.com/ibm/history/
[33] About IBM: http://www.ibm.com/ibm/us/
[34] IBM Products and Services: http://www.ibm.com/products/us/
[35] IBM's developer-dedicated site: http://www.ibm.com/developerworks/
[36] WebSphere product family: http://www.ibm.com/software/websphere/
[37] WebSphere Portal: http://www.ibm.com/software/webservers/portal/

7.4.1 Historical Review of the Portlet API

To better understand the link between Apache Jetspeed and IBM WebSphere Portal and the shared Portlet API, a historical review is required. The information presented in this subsection is mostly obtained from the Jetspeed mailing list archives[38]. A couple of helpful persons still working on the Jetspeed project, which were on this project at the time of these discussion, have also given valuable information[39].

The Jetspeed project were started in 1999, confer the Jetspeed analysis, Chapter 7.1 – "Apache Software Foundation – Jakarta Jetspeed". At the same time, IBM had a portal system. Early on, several IBM employees joined the Jetspeed mailing lists, and began to collaborate and contribute code to the Jetspeed project. Of special interest are the names Ingo Schuster, Thomas F. Boehme, Thomas Schaeck and Stephan Hesmer. In several aspects, it can look like these IBM employees have run the entire Portal API development within IBM, and used Jetspeed and its Open Source community to drive the discussions around, and the development of this API.

New Jetspeed Portlet API proposal

The 15th of November 2000, Thomas Schaeck put forth a document called "Portlet API Requirements" after some initial discussion on the Jetspeed list. The first and most important requirement in this document is that the Portlet API should be "self contained". This would enable other vendors to implement the API, without the heritage from Jetspeed and Turbine. It would also enable a portlet written for one vendor's portal to be moved to another vendor's portal, providing that both implemented the Portlet API.

Some time after, around December 2000, the JavaDocs and the Java source code for the proposed API was added to the Jetspeed CVS archives by Ingo Schuster. Thomas F. Boehme authored most of this code, and Stephan Hesmer did the rest. During the next couple of months, several discussions were held, both at the Jetspeed mailing lists, and at an IRC chat that were organized. IBM gathered input from many people from several different companies during this process. The set of documentation and sources were updated several times during these months, mostly by IBM employees.

At about March 2001, the Jetspeed community, the mailing list participants, including persons from several other corporations, and IBM, finally came to an agreement over the design of the API.

Some time after these discussions, IBM released the WebSphere Portal Version 1.2, which implemented the Jetspeed API, somewhat augmented (for example, a "service" package were added). Later, a follow-on release, version 2.1, was released. [IBM B]

December 25, 2001[40], a document called "Portlet API – First Draft" [IBM C] was added to the Jetspeed CVS repository. This document was written by Stephan Hesmer, Stefan Hepper and Thomas Schaeck, all at IBM.

---

[38] Old Jetspeed mailing lists (hosted at Working-dogs) http://www.mail-archive.com/jetspeed@list.working-dogs.com/
New Jetspeed developers mail list (hosted at Apache) http://marc.theaimsgroup.com/?l=jetspeed-dev&r=1&w=2

[39] Specifically, Raphaël Luta and Glenn Golden have kindly answered the authors questions.

[40] This is Christmas Day (!). However, the date seems to be correct, as it says so both in the CVS logs and in the PDF-file.

Java Specification Requests for Portlet API

Early in January 2002, IBM submitted a Java Specification Request for the creation of the "Portlet API" specification. This became JSR 162 [JSR 162]. The contact person for the submission was Thomas Schaeck, and the specification lead was Stefan Hepper. Less than one week later, Sun Microsystems Inc. also submitted a JSR for "Java™ Portlet Specification", which became JSR 167 [JSR 167].

Before any of the specification's review time had elapsed, both JSRs were withdrawn on January 20, 2002. A very short time after, the JSR 168 "Portlet Specification" [JSR 168] was created, with IBM and Sun as the submitting members, and with all information from the JSR 162 and JSR 167 combined into one submission. This JSR's *Review Ballot* deadline was on February 11; most members of the Executive Committee for SE/EE approved the ballot, and none was against it.

The timeline set for the standardization process is rather short, with *Community Review* in May, *Public Review* in July, *Final Draft* in October, and implemented *Test Compatibility Kit* (TCK) and a *Reference Implementation* in December 2002. The reference implementation is supposed to be developed in an Open Source fashion, at Apache, lead by IBM[41].

IBM WebSphere Portal version4.1

On April 2, 2002, IBM released their first edition of the "Portlet Development Guide – Working with the Portlet API 1.1" [IBM A]. This is a document associated with the WebSphere Portal version 4.1, due to be released in May 2002, with general availability in June 2002.

This document outlines a further developed Jetspeed API. This API peculiarly is not present anywhere in the CVS repository for the Jetspeed project. Nevertheless, it is a very interesting API, with a high level of functionality. This document is written by Stephan Hesmer and Ingo Schuster, in addition to two other persons not mentioned earlier. In the overview chapter of this document, the authors state, "The Portlet API offered in the WebSphere Portal Version 4.1 is the first step toward the Portlet API standardization". This seems as a good indication of what IBM will bring to the standardization table.

Because this is one of the most developed portlet APIs around, and because it has been developed with help from multiple organizations, it API is likely to influence the Java standardization process which is in progress in a considerable degree. The WebSphere Portal is therefore deemed interesting enough to review in spite of the current lack of an actual product.

7.4.2 Architecture and Architectural Features

The "Portlet Development Guide" shows that several changes are made to the Portlet API since the last revision that is included in the Jetspeed CVS archives and the version included with the WebSphere Portal version 2.1. Whenever these APIs are compared, the one included in the WebSphere Portal v4.1 is referred to as the "IBM

[41] Sun did this with the Servlets specification 2.2 and 2.3, where the reference implementations, Tomcat 3.2 and Tomcat 4.0, were developed at Apache Jakarta.

API", while the one contributed to the CVS repository to Jetspeed, which later were implemented in WebSphere Portal v2.1, is called the "Jetspeed API".

J2EE Application Server Requirement

Most of the two APIs functionalities are the same. However, there is one major difference: while the Jetspeed API were completely stand-alone, expecting a implementing server, the so-called *Portlet Container*, to handle the life-cycle of the portlets, the IBM API defines a portlet to be an extension of a *Servlet*, thus requiring a *Servlet Container* to handle the portlet instantiation and initiation[42].

The Jetspeed API had a concept of a *Portal Application*. This was a set of portlets, bundled with the deployment descriptions for the portlets, into a single file. Such *Portlet Application Archives* (with the extension ".par") could be dynamically loaded and unloaded into the portlet container by the administrator. This has in the IBM API been replaced by the use of the J2EE concept *Web Application*. This means that a portlet application is bundled as a *Web Application Archive* (with the extension ".war").

The result of this is that if the basis of IBM's API is going to be the foundation for the Portlet API standard, then a portal server cannot any longer be simply a servlet container, but must be a full J2EE implementation.

"Evolution"

In the "Portlet Migration Guide" [IBM B], the author explains the API's dependency on J2EE as being an evolution towards a standardization of the Portlet API. Some highlights from this evolution are both more conformance towards the J2EE specification, and a tighter integration with the application server. How the integration between the different portlet web applications should be done, how the portal itself is supposed to be implemented, and how the loading and unloading of the portlets is done is not elaborated on in any of the documents available. Since the system is not available yet, further aspects of the architecture are difficult to deduce.

Clean API

The API and its methods are made in a vendor-independent fashion, and is mostly an extension to the already existing, well-defined Servlet API version 2.3. In effect, it is a way for a web page to be constructed from multiple small java classes, portlets, working independently, instead of the entire web page being constructed by one piece of program logic; the servlet. This will be reviewed more thoroughly in the next subsections.

Relationship and similarities with the Servlet API

Both the Jetspeed API and the IBM API are similar to the Servlet specification. For the Jetspeed API, the reason for this was more "cosmetically" than a necessity; the API should be easy to learn for people already familiar with the servlet API, and it should be easy to port a servlet to the new portlet environment. For the IBM API, the servlet similarity is more of a requirement, since the Portlet API is a direct extension of the Servlet API.

[42] Confer with Appendix B for a review over the different aspects of the J2EE specification.

7.4.3 Framework concepts

Since the actual server implementation is not available, and the available documentation is only the two documents already referred to, it is a bit difficult to analyze the framework concepts. For example, the portal is probably using the administration module available with the WebSphere application server. However, some of these concepts are revealed from the API, and these will be reviewed.

Multiple Pages

The workbenches referred to in this API is called pages. The portlets are placed on pages, and user and group persistent data is scoped to the level of a page. There are two different types of pages: user pages and group pages.

Configuration on multiple levels

The portlets may be configured by the portlet developed, using the servlet specification's normal deployment descriptor (web.xml). Each portlet instance is also configured in the portal specific deployment descriptor (portlet.xml). In addition, there are two run-time configuration possibilities. This is explored in detail in the next section.

Portlet Modes and States

The portlet registers support for different modes, for example help and edit. This is configured in the portlet deployment descriptor. The portal server then displays icons for these modes, and invokes the respective mode switching action on the portlet if the user clicks on any of these. A portlet on a user's page may exist in three states: normal, maximized and minimized.

7.4.4 Framework features

As the API is too extensive to cover thoroughly here, only an overview will be presented. The full description of the API may be found in the "Portlet Development Guide" [IBM A] and the JavaDocs for this new API will most likely be available at IBM's website shortly.

Portlet

The Portlet interface is an extension of the Servlet interface found in the Servlet API. The relationship between the Servlet API, the Portlet API and an independently developed portlet "ThePortlet" can be seen in the following relationship:

```
javax.servlet.Servlet
 \---javax.servlet.http.HttpServlet
      \---org.apache.jetspeed.portlet.Portlet
           \---org.apache.jetspeed.portlet.PortletAdapter
                \---com.some.company.ThePortlet
```

By extending the PortletAdapter class instead of implementing the Portlet interface, one is alleviated of implementing every single method the Portlet interface describes. In addition, this is a common "trick" to enable the Portlet interface to evolve, without the developers needing to change all their portlet code to include the new methods that the evolved Portlet interface describes.

The servlet dependency requires that the portlet must be defined in two different deployment descriptors. First, it must be defined as a servlet, so that the servlet container is aware of it and may load and initialize the code. Secondly, the portlet must be defined in a separate file so that the portal system may invoke the code. The different definitions have different scope, which will be made clearer in the next subsection.

Class Instance, Concrete, Concrete Instance and User Instance

As with the Servlet specification, only one portlet is ever physically *instantiated*. This is called the *portlet class instance*. A *Concrete Portlet* is a portlet parameterized by a single PortletSettings object. Such concrete portlets are created whenever a portlet application is installed, since a portlet deployment descriptor must contain at least one concrete portlet for each portlet defined. In addition, an administrator may create multiple concrete portlets by configuring a portlet, thus giving it multiple PortletSettings objects, possibly containing different parameters.

When a user places a portlet on one of his pages, or an administrator places a portlet on a group page, a *Concrete Portlet Instance* is created. This is a concrete portlet parameterized by a single PortletData object. The PortletData object stores persistent data about this single instance residing on a specific page. The PortletData object's parameters is only to be changed by the portlet itself. The relationships between these different entities are shown in Figure 26.

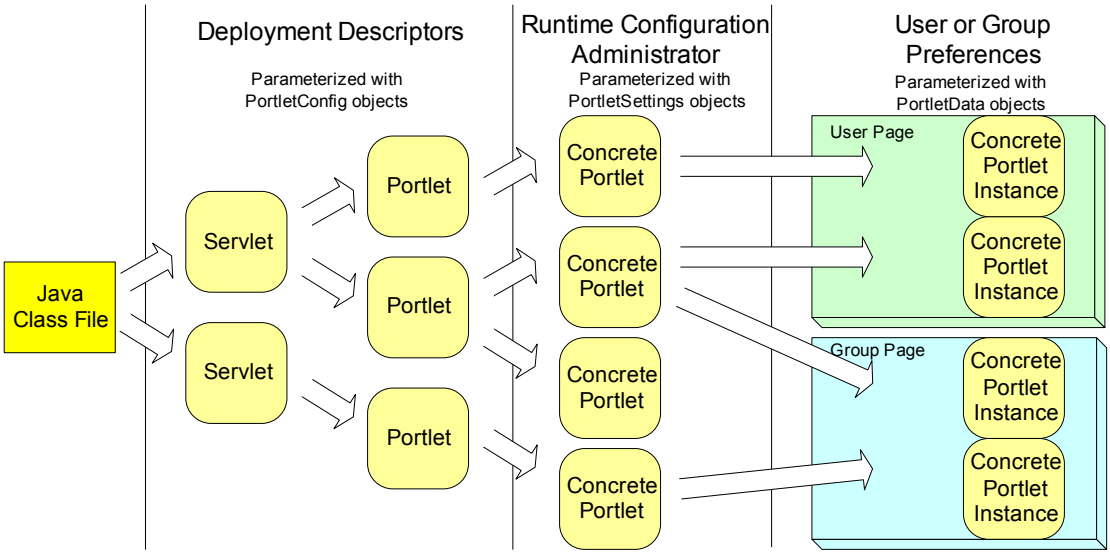


Figure 26 Relationship between the different levels of instantiation in the IBM Portlet API

Finally, when the user actually views the page, a PortletSession object is created for each concrete portlet instance the user have on this particular page. This object represents the parameterization of the User Portlet Instance, and stores transient information regarding this single invocation of the portlet.

Portlet Applications

As mentioned, the portlets can be grouped into *Portlet Applications*. A portal application will typically be a specific module, for example, a group calendar system or a project planning system. The portlet application contains all resources needed to

run the portlet application, for example, the images, property files, and active components like JSPs, Java code and the portlets (Please confer with Appendix B – Web Application). A portlet must be packaged within a portlet application.

In the same manner as with the portlet and concrete portlet relationship, a concept of portlet application and *Concrete Portlet Application* is employed. A concrete portlet application is parameterized by a PortletApplicationSettings object, and there may exist several of these. A concrete portlet application contains at least one concrete portlet from the portlet application, but does not need to contain all of them.

Built-in Caching

The Portlet API facilitates server-facilitated caching. This is done by letting the Portlet implement a getLastModified() method, which should return the milliseconds since the Epoch[43]. The portal may use this value by checking it against the value returned the last time this same portlet was rendered. If it returns the same value as last time, it can skip the rendering stage of the portlet, since it apparently has not been modified. A special value signifies that the portlet does not want caching, thus the portal server should always render the portlet fully.

Listeners

The portlet may implement a set of listeners to be notified of certain types of events. The PortletSessionListener interface makes the portlet aware of users that establish a session, which in most circumstances means that the user logged in. It also notifies all portlets of a user shutting down a session, which happens on logout or timeout.

The PortletPageListener gives the portlet a chance to output data to the output stream before any of the servicing methods are being called on any portlet. In addition, the portlet may output data to the stream after all service methods for all portlets are invoked. This can be used to output some JavaScript code that should be accessible to all user portlet instances.

Finally, the portlet may implement PortletTitleListener to enable the title to be dynamically changed, based on some decision criterion. The title could for example be tailored to whether the portlet showed a detailed or overview description of some customer, or be changed according to which type of client that has connected.

Events system: Window, Action and Message events

The Portlet API makes use of the event concept inherent in several of Java's APIs. If a user clicks a button on the browser, or submits a form or similar user activities that the portlet has registered actions with, an ActionEvent is fired to the appropriate portlet, notifying it of the occurred action. If a user clicks on any window control button of the portlet window, for example "maximize", the portlet gets a WindowEvent fired, describing which type of window event that happened.

The sending of messages to other portlets is facilitated by using the send-method available in the PortletContext. This method enables the transportation of an object implementing the PortletMessage interface to another portlet. The server passes the message by invocation of a MessageEvent on the receiving portlet.

[43] The Epoch is "the beginning of time" for a UNIX system, and is defined to be midnight, January 1, 1970, UTC.

This event style of programming is similar to the one use in the Java Foundation Classes, and makes developing portlets similar to developing any ordinary Java based application with a graphical user interface.

Direct Access to output stream

All the service methods that let the portlet output anything, is providing direct access to the output stream. This is a very simple and powerful abstraction. However, if one portlet crashes or delivers faulty markup, the whole page is affected. This is a volatile solution, where each portlet must work perfectly to enable the portal to function properly.

User Abstraction Object

From the PortletRequest object, passed with the invocation of any rendering methods, the portlet developer may get hold of an object representing the currently authenticated user. The User object contains information about the user, like the nickname, the full name, the last login time, and the user's ID.

Multi Markup

The API supports multiple markup types. Each portlets tell the portlet server which markups they provide in the portlet deployment descriptor. This only makes the different portlets available to be configured onto the workbenches for the different user agents. It is up to the portlet to actually check which client that is connected or which markup the connected client supports by querying the PortletRequest object, and then supply that markup.

Configuration and State

As mentioned in the previous subsection, there are several levels where configurations and customizations may occur. Initialization parameters are available using the PortletConfig object, which is an extension to the ServletConfig object. These parameters are global to all concrete portlets created out of this portlet, while the PortletSettings object contains parameters specific to each concrete portlet.

As an example, consider an administrator that has acquired a news-fetcher portlet. He makes one *concrete portlet* to fetch the headlines from BBC, and another concrete portlet to fetch news summaries from the CNN site. A list over supported new source formats and their supporting URL-fetcher class files are contained in the PortletConfig object, thus shared between both these two concrete portlets.

A group administrator places a concrete portlet on a group page. This creates a *concrete portlet instance*, and an associated PortletData object is instantiated. The group administrator configures the concrete portlet instance of the BBC news portlet to display the news in terse format, while the CNN instance should be displayed in full format with thumbnail picture. These settings are stored in the PortletData object, and are persistent.

Finally, when a user actually views this page, PortletSession objects are created. These are used to keep the state of the portlet when the user clicks on the "next ten" or "previous ten news articles" buttons, thus changing the current view of the portlet. For the portlet to remain on the chosen page of articles during a reload of the page, the

selection is kept as a parameter in the PortletSession object. When the user logs out, and then logs in again, the PortletSession object is destroyed, and the selection will thus be reset to show the last then articles.

Client, Capabilities and Multiple Markup Languages

Upon invoking the methods in the Portlet interface, the PortletRequest object may be queried for which type of client is connected. The returned Client object contains information about which user agent is connected, which markup or markups that are supported, which mime-types that are supported, and which capabilities the client support. Capabilities include level of HTML, whether JavaScript is supported and so on. This enables the developer to tailor its output for different clients, and may do this based on different levels of functionality supported by the client.

Name-clash Avoidance

Since there can be several portlets on one page, and even several instances of the same portlet, a definite possibility of name-clashes arise, in addition to the addressing problem of how to specify that a argument should be directed towards a specific portlet. This is solved by letting a portlet runs within its own unique namespace. It may use one of several methods to get hold of its current namespace and ID. This is used to qualify attribute names, JavaScript function names and global variables in the output of the particular portlet.

7.4.5 Software Quality Model References

Fault tolerance

The IBM system suffers from the same problem as the BEA system in regard to fault tolerance; if one portlet crashes during rendering, the entire portal workbench might crash. This stems from the method the portlets have to output their contents; the portlet is given a direct handle to the output stream. If the portlet writes half of its output, and then crashes, the resulting page might very well contain inconsistencies.

Changeability

The stand-alone API gives developers a clear and consistent way of making new portlets, however, it does not enable the developers to change anything regarding design and rendering of workbenches, including the traditional theme support, layout changes and similar concepts. If the system provides for such features, this is defined outside of the described API.

Testability

A stand-alone API makes testing of single units much easier. Since the API is detached from the actual server implementation, a special testing harness may implement it, and in effect run the portlet. The portlet can thus be tested for conformance with the API contract. However, IBM does not mention such a testing harness, but the API detachment makes this possible.

Adaptability

The IBM API includes a method for accessing localized text from a resource bundle. However, the user properties object does not contain any attributes denoting which

language the user prefers, and the API does not contain any helper methods for localizing dates, times, number or currencies either.

The IBM API defines a Client object on which the portlet may query which markup or markup languages are supported. In addition, the client object can tell the portlet what capabilities the client support, for example which level of HTML and JavaScript it can handle.

Conformance

The IBM WebSphere Portal system specifically requires the IBM WebSphere J2EE server, and thus does not conform to the J2EE standard. However, the IBM WebSphere J2EE server conforms to the J2EE standards, thus being able to run components conforming to these standards.

IBM's Portlet API holds some promise of influencing the Java standard Portlet API to some degree. This because the API has already been through some peer reviewing on the open Apache Jetspeed list, and because it is very open in nature, conforming to already existing standards, for example J2EE. In addition, IBM is co-spec-leads for the standardization process, thus might have some additional influence on the final specification. However, 20 different companies, every one with their own ideas of how the final API should look, are jointly developing the Portal API standard. The final result might deviate quite some from the IBM API.

7.5 Oracle – Oracle Portal

Oracle boasts to be "… the world's largest enterprise software company, providing enterprise software to the world's largest and most successful businesses"[44]. With 2001 revenues exceeding $10.8 billion, the software giant are used by 98 of the Fortune 100 companies. The company has more than 42.000 employees. The product line includes database, application servers, application development tools, enterprise business applications and decision support systems.

On their "The Oracle Story" page[45], the business idea that Oracle was founded on is explained. Twenty-five years ago, the Oracle Chairman and CEO Lawrence J. Ellison came across a description of a relational database prototype. He saw the importance and possibilities that lay in this technology, and realized that no company had committed to commercializing this type of system. On these grounds, he founded the successful company in 1977, with two co-founders, Bob Miner and Ed Oats.

In 1995, Oracle decided to commit to Java and in 1997, they began to implement J2EE standards throughout their product line. Oracle have made a document entitled "J2EE and Microsoft .NET" [Oracle B], where they reason that open standards, portability and choice weighs in favor for J2EE, compared to "the one-vendor proprietary approach of Microsoft .NET."

[44] About Oracle: http://www.oracle.com/corporate/
[45] "The Oracle Story": http://oracle.com/corporate/index.html?story.html

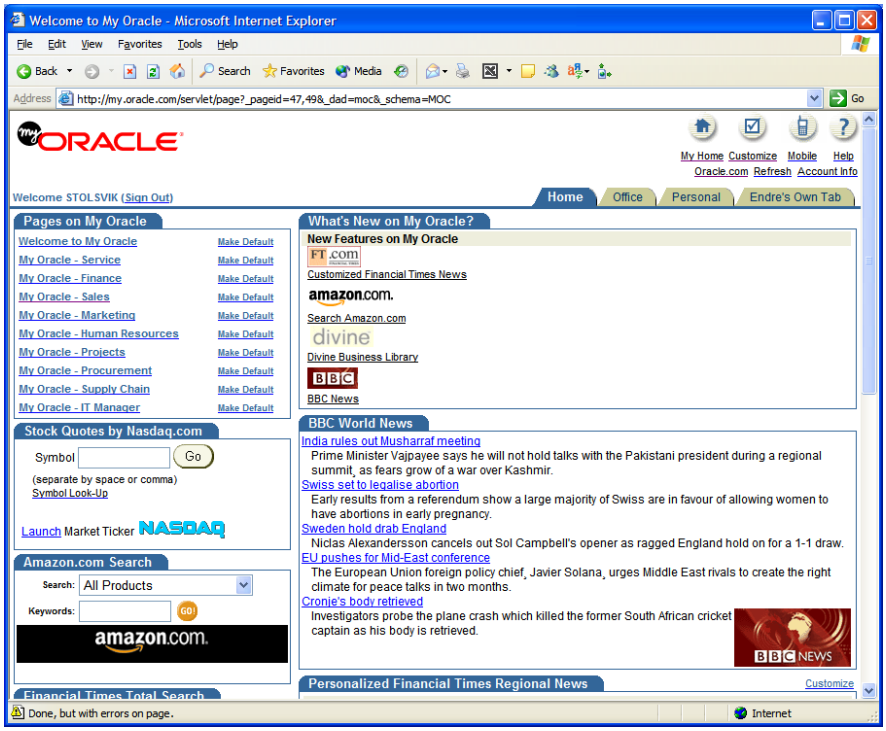


Figure 27 Screenshot from Oracle Portal (from MyOracle @ Oracle)

7.5.1 Architecture and Architectural Features

Oracle 9iAS Portal is built upon the Oracle 9i Application Server's core services [Oracle C] and is itself a service. The integration is "complete", as the portal service is an integral part of the application server. The portal server is available for both the Oracle 9iAS Standard and Enterprise Edition. In addition to the application server, the portal also requires the Oracle 9i Database. Figure 28 shows the architecture of the portal.

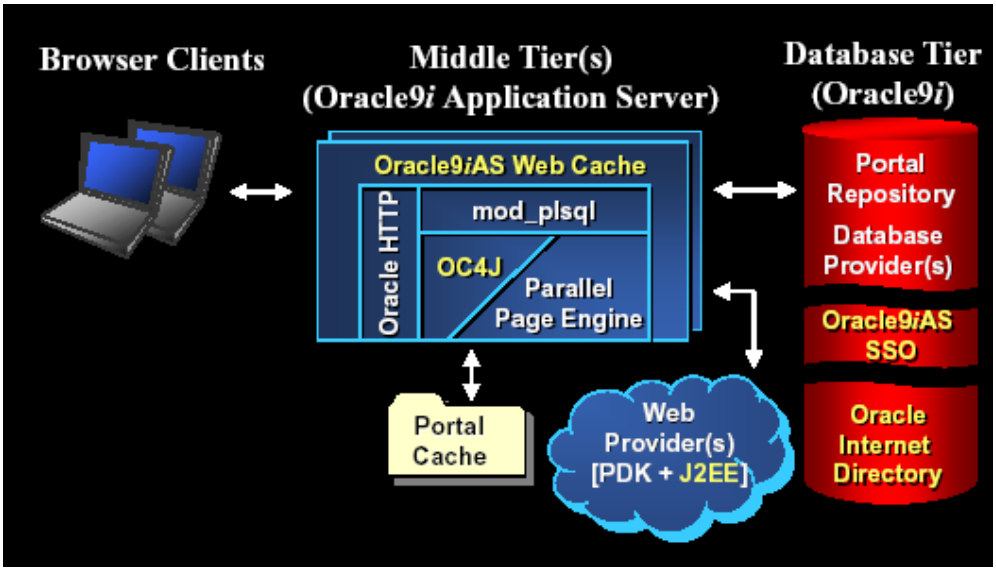


Figure 28 Oracle9iAS Portal Architecture (From [Oracle C])

Throughout all their documentation, Oracle is very anxious to explain that the portal exclusively is "Built upon open internet standards". Against these types of statements, it is somewhat peculiar to note that the Oracle portal server requires both Oracle's application server with its host of services and Oracle's database, thus completely locking the customers into Oracle's particular architecture.

Oracle 9i Application Server

Oracle's "Application Server" product line is the Apache HTTPD Web Server, extended with a set of "mods". The Apache server is modular in nature, and all functionality except the raw delivery of files are facilitated by such mods. Oracle supplies the following modules:

- mod_fastcgi for persistent CGI processes
- mod_plsql for direct invocations of PL/SQL procedures stored in the Oracle database
- mod_oradav for distributed authoring and versioning support
- mod_ossl for SSL security implemented by Oracle
- mod_osso for routing of Oracle9iAS single sign-on request
- mod_oc4j for communication between the Apache processes and Oracles J2EE implementation, OC4J.

OC4J

Oracle9iAS Containers for J2EE, OC4J, implements the Servlet Container and EJB Containers for the J2EE specification [Oracle D]. Oracle frequently emphasizes that this is an "extremely lightweight and fast" container. In addition, Oracle frequently mentions the JServ product from Apache, which is a Servlet Container implementing the Servlet 2.1 standard. This server is from Apache's side now deprecated in favor for the Tomcat line of servlet containers.

The Portal Server

The portal is an integral part of the Oracle9i Application Server. Judging from the "Technical Overview" white paper [Oracle C], the assembling of pages and portlets is done by a core portal engine, which is built upon the OC4J. The portal engine is thus a Java Servlet based system. However, it is tightly integrated with the other services in the application server suite.

Text-based Configuration Files

Much of Oracle's systems and subsystems use text based configuration files, similar to how most of Unix' systems are configured. Several of Oracle's Application Server systems are derivates from Open Source projects, specifically Apache Web Server and Apache Java JServ. Oracle's newer OC4J is using the same configuration setup that Apache Tomcat is using, making a developer from the open source community feel comfortable with the system. Oracle also supports Apache Ant, a Java building system.

Portlet Providers

A portlet communicates with the portal via a *Provider* [Oracle C, E]. Each portlet must be connected to exactly one provider, but one provider may contain multiple portlets. The portlets of a provider exposes its underlying application or information

sources. When the portal needs information about a particular portlet, or needs a portlet to be rendered to a web page, it communicates with the provider.

Providers can be used for grouping of portlets into functionally groups. In addition, provider groups provide for an extra level of aggregation. Providers may be developed in several ways, from programmatic to declarative. There are broadly two different types of providers: *Web Providers* and *PL/SQL Providers*.

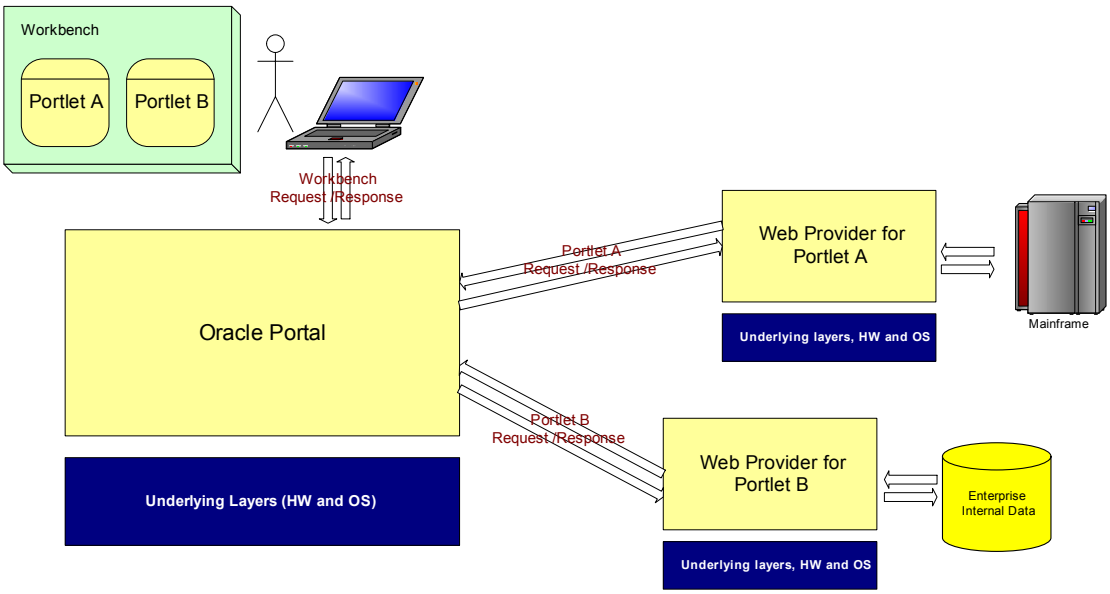


Figure 29 Oracle Portal, connecting to Web providers, connecting to backend data

Communicating between the web provider and the portal is done using SOAP remote procedure calls over HTTP. A remote web server does not have to be Oracle's; it can be any application server. This enables the provider and portlet logic to be made in any programming language, as long as it input and output may be exposed over HTTP. This setup is shown in Figure 29.

The output of such a portlet may optionally be run through a XSL transform. To alleviate the portlet developer for the burden of the somewhat complicated remote procedure calls and transactions that is needed for the portal, the provider and the portlet to communicate, Oracle have made a set of Portlet Development Kits, PDKs. These PDKs hides this complexity behind a set of APIs. Oracle has made a PDK for Java development and one for PL/SQL. In addition, the May 2002 release of the PDK contains beta-versions of a PDK for URL based portlets and a PDK for Web Services based portlets.

This remote approach is very similar to the main idea of the Plumtree portal system, which is analyzed in the next subchapter. The general layout of this remote approach is described in more detail there.

The PL/SQL Provider must be installed on an Oracle9i Application Server, not necessarily the same as the portal server. Using this type of provider, long-time customers of Oracle may leverage their existing knowledge of the database procedural

language PL/SQL to develop portlets. This approach features direct access to the database on which the PL/SQL Provider is running.

The providers are registered using the administrative interface in the portal. When doing so, the portal connects to the provider and queries which portlets it has available. When the provider replies, the portal incorporates the portlets into its registry, making them available to be placed on portal pages. Per default, only the user that added the provider have access to the portlets, so he must alter the access control list for the portlets manually to some more lenient restrictions.

7.5.2 Framework concepts

Internationalization, Multiple Languages

All text appearing in the portal, including wizards, dialog boxes and help files, are translated to 27 different languages. The language selection is based on the user's browser settings. The self-service publishing facilities within the portal also enables users to publish their content in multiple translations.

Authentication

The portal uses the application server's Single Sign-On (SSO) architecture for authentication. Using this system, a user may log on to any part of any system developed on the application server, and gains access to any resource protected by the SSO system. The SSO is fully integrated with Oracle's Internet Directory (OID), but can also integrate towards third party vendors' security management systems. Netegrity's SiteMinder is mentioned as an example.

Users and groups

The Portal uses Oracle Internet Directory as its repository for user and group definitions. The user and group structure may be administered through the built-in administration interface embedded in the portal, or using the administration tools and APIs delivered with the OID system.

Access Control Lists and Administration Delegation

An access control list (ACL) system is employed throughout the portal server. Access control lists protect portal objects, including pages, styles, items and portlets. Privileges for accessing, customizing or modifying the different objects are granted to specific users and groups. The portal administrator may delegate ACL responsibility to object owners. An administrator may also grant global privileges to sub-administrators on all objects of a given type.

Administration of the Portal through the Portal

The portal is itself fully portlet enabled, and the portal is preconfigured with a set of pages and portlets for portal development and administration. Here the users and groups may be managed.

The portal system have import and export features providing for staging and export of portal content. This includes the page definitions, security settings and all item and portlet content. The administrator may export from one or several development servers and import the definitions onto the production site.

The Oracle9i Application Server includes a system called the Oracle Enterprise Manager (OEM), to which the portal system is integrated. Using the user interface of this system, an administrator may monitor and maintain the data, portal configuration files and services underlying the portal system. This includes the HTTP, mod_pl/sql, and web cache services, in addition to the servlet engine, the portal database, the single-sign on system and the portlet provides.

Portal Navigator

The portal navigator is per default available from the portal page banner. It enables administrators and users to browse the different entities in the portal system, including the pages, providers and database objects. The user can locate a specific entity, and may perform actions on that entity, both operations constrained by the user's privileges to the entity in question.

Hosting, Application Service Providers and Virtual Private Database

A company's portal system can be hosted on a shared server that is hosting several other companies' portals. This is done by using Oracle9i database specific feature called *Virtual Private Database*. This system adds one more dimension to the database's tables, isolating the different companies' data from each other, while still using the same database tables.

This enables a hosting environment to keep several customers' portals on one portal server and database installation, instead of having one portal and one database for each customer. This is a cost-effective solution, since a high number of portal servers and database servers would require a significant number of physical servers. In addition, the administration overhead of managing a great number of database and portal installations is greatly reduced.

In such an environment, a user logs in with not only his username and password, but must in addition supply the company name. Alternatively, the system can be configured to allow for *Branded URLs*, which enable one company to access their portal at http://portal.companyA.com/, while another company accesses their portal at http://portal.companyB.com/. The company is called a *Subscriber* of the portal when hosted in this way.

A setup script facilitates the instantiation of a new portal on an already existing installation. This makes an administrator able to setup such a subscription account in the matter of minutes, instead of the hours needed installing a full portal system. The script takes care of all the initial data entries in the database and authentication system, and includes the setup of default portal pages.

7.5.3 Framework features

Portlet Development

Oracle's portal system enables developers to create portlets in three different ways:

- Declarative: The portal enables the developer to create portlets using a wizard-driven, declarative interface. Such portlets may contain content published through the portal content management system. There are also wizards for building forms,

reports, charts and other data-driven portlets that may communicate directly with the Oracle database, as well as other databases.

- Programmatic: By use of the Portlet Development Kit, a developer can program portlets in PL/SQL, Java and other web-enabled development environements.

- Oracle Tools: The Oracle Reports 9i, Oracle Forms 9i, Oracle Discoverer 9i and JDeveloper are all integrated with the portal through the use of Single Sign-on and automated features and wizards for creating portlets.

Portlet Development Kit

The portlet development kit alleviates the developer of the complicated communication between the portal, the provider and the portlet, hiding the intricacies behind simpler Java APIs. The Java PDK operates by translating the remote procedure calls from the portal to normal method invocations on Java interfaces. However, the PDK from Oracle is somewhat chaotic, with lots of classes referring to each other and several of the API's methods lacking documentation. It may seem like the PDK has been through a couple of generations without a complete overhaul. Compared to the clean APIs of the "new Jetspeed API" and the IBM API, Oracle's API falls through as unstructured.

A central element in the PDK is a Servlet that interfaces between the portal and the provider. The developer is expected to configure his servlet container himself, providing the necessary configuration parameters to enable the servlet container to load this servlet. Oracle points out that this servlet may run on any servlet container, including BEA WebLogic, IBM WebSphere or Apache Tomcat.

Portlet Definition

A portlet is implemented by defining as set of controllers: a renderer, and optionally a personalization manager and a security manager. These definitions are registered in a XML file. This XML file is read by the provider within the PDK on servlet container startup. When the portal asks which portlets the provider provides, it replies with this list.

Access Restriction

The PDK have facilities to enable the developer to restrict access to the portlet in several ways. It can control whether a user should be able to see the portlet at all. In addition, the portlet can decide whether to enable or disable certain functionality, based on the user Authentication Level. This latter feature seems a little crude; the return value of the authentication level getter is simply an integer.

User abstraction object

A *User* object gives access to the user using the portal. This might be a "PUBLIC" user, denoting that the user at the other end has not logged on yet. If the user has logged on, the user object enables the developer to get parameters of the user, including username, the *Subscriber* object and a *Location* object for the user. The subscriber is usually a company; confer the subsection on hosting and ASP environments. The location includes the address of the individual and his company, and strangely enough a getter for the user's X and Y position.

Preferences

User preferences are stored locally on the server running the portlet, using either a DBPreferenceStore or a FilePreferenceStore. An obvious problem here is the question of what happens when the user is deleted from the portal. There does not seem to be any automatic deletion of preferences and personalization data from the database or file system on the server running the portlet. This contrasts to how Plumtree have solved this problem, as they have a mechanism of sending the customization data back to the portal server for storage.

7.5.4 Software Quality Model References

Adaptability

Oracle boasts that their portal is internationalized. However, the only element mentioned is the language selection, and again the dates, times, currencies and numbers are not mentioned. The locale is based upon the browser setting.

Conformance

The Oracle portal system requires both the Oracle application server and the Oracle database. This makes it one of the most vendor locking portal systems around. However, the Oracle 9i application server conforms to the J2EE standards, thus being able to run components conforming to these standards.

Scalability

The Oracle 9i Application Server is scalable, and can run on multiple physical servers. The portal system inherits this feature, and it is thus possible to handle very many concurrent users.

7.6 Plumtree – Plumtree Corporate Portal

Plumtree prides themselves of being "the founder and leader of the corporate portal market". Incepted in 1997, the company definitely was one of the first to enter the market. They claim to have been cash-flow positive since 2001, and grown revenues consistently since their inception. They are a dedicated corporate portal business [Plumtree G].

The company states: "Unlike almost any other type of software, corporate portal software is designed to integrate rather than replace customers existing technology". Based on this, "Plumtree Software has sought to develop the most extensive partnerships in the industry". They have *Channel Partners*, being partners that deploy and roll out portals to customers, and *Technology Partners*, which are hardware and platform vendors, as well as software companies that focus on a large set of business software [Plumtree F].

Plumtree have over 300 customers, and 61 of these are on the Fortune 500 list. The customers combined, the portal has five million licensed users[46].

[46] About Plumtree: http://www.plumtree.com/company

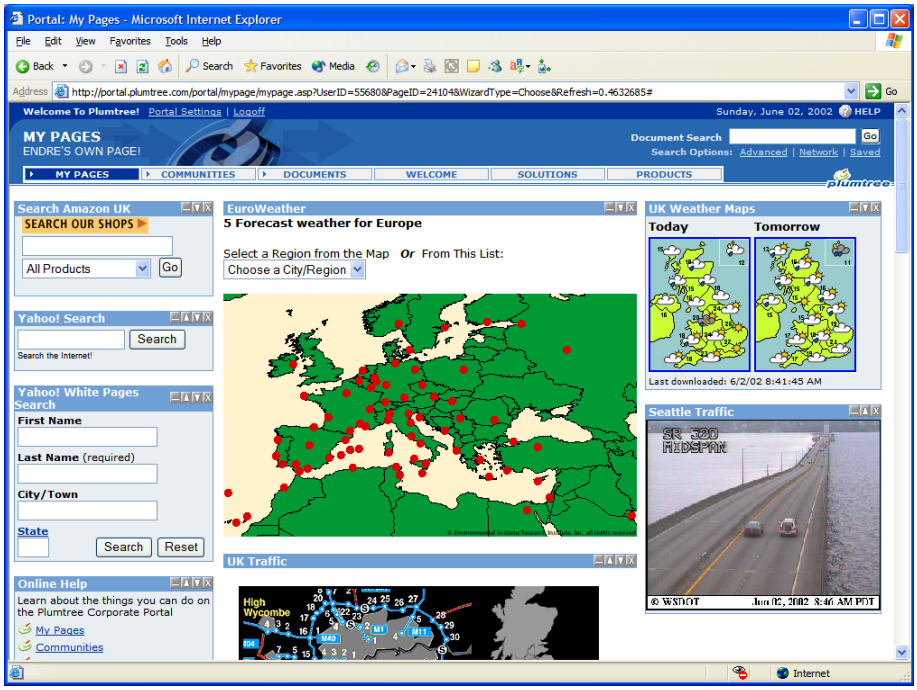


Figure 30 Screenshot from Plumtree Portal (from demo site @ Plumtree)

7.6.1 Architecture and Architectural Features

The server is only available for the Windows NT/2000 and the Sun Solaris platforms. This suggests that the lower levels of the architecture are programmed in some natively compiled language, for example C or C++, and not Java, as many other portal vendors have chosen.

The Plumtree portal system features a two-tier server, whose structure they call the *Internet Architecture*. This is an interesting portlet rendering approach, where the rendering of the portlet workbenches are split in two simultaneously working stages. The portlets of Plumtree's architecture are called *Gadget Web Services*, and are hosted on dedicated portlet servers, *Plumtree Gadget Servers*. Note, though, that this has nothing to do with the newer term *Web Service*, whose technology is based on XML (See discussion of this in [Plumtree D, E] and [Plumtree B], Appendix A). The portal server is called *Plumtree Portal Server*, and serves as the hub in this architecture.

To render a workbench, the user's browser communicates with the portal server. This server in turn figures out, based on the user's layout of the workbench, which portlets it needs to fetch. Requests are sent out to each server hosting the relevant portlets, asking them to render the portlets. This is done in parallel, using Plumtree's *Massively Parallel Portal Engine* (MPPE). If the portlet has enterprise application integration functionality, the gadget server may be hosted on the same physical server as the enterprise application, or it may be hosted on a separate physical server. This makes the process three-stage. The portal server finally collects all the separate rendered portlets, assembles the completed workbench and outputs it to the user. This process is shown in Figure 31.

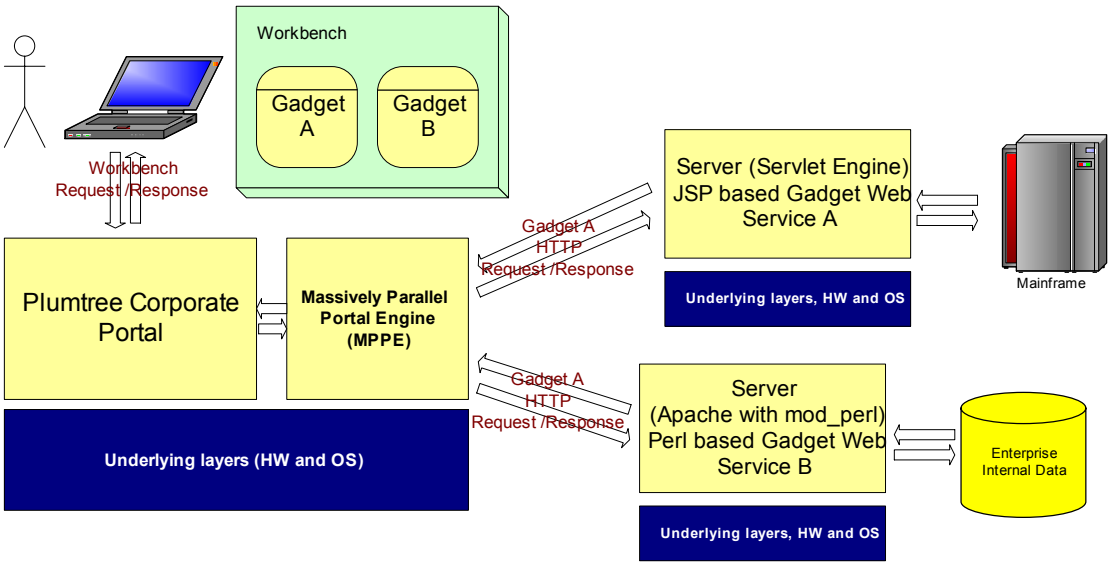


Figure 31 Plumtree Architecture, Massively Parallel Portal Engine

The communication between the portal server and the gadget web service server is done using ordinary HTTP requests over TCP/IP. This makes almost anything capable of being a gadget web service, as long as it can read HTTP and reply using HTTP headers and HTML or XML. This means that portlets for the Plumtree system may be developed using any HTTP aware platform, it being Microsoft's ASP, Macromedia's ColdFusion or the Open Source project Perl running via CGI or inside the Apache HTTPD web server. Plumtree have developed code packages that enable rapid development cycles. These are called gadget development kits, and are currently available for Active Server Pages (ASP), Java ServerPages (JSP), Java Servlets, Perl and mod_perl.

The preferences details for a gadget web service are supplied to the gadget server using *HTTP headers*. The initial preferences configuration is initiated when the portlet notes that it doesn't get the preferences HTTP headers sent to it when it gets a request. It therefore shows its configuration screen instead of the gadget content. When the user replies, the gadget web service sets up a set of variables and tells the portal server, via the returning HTTP headers, to store the preferences in the user's profile. The next time the user requests the same portlet, the server sends along the preferences variables to the gadget server. The user may request to the gadget that it displays the configuration screen again, giving new configuration details. This process is shown in Figure 32.

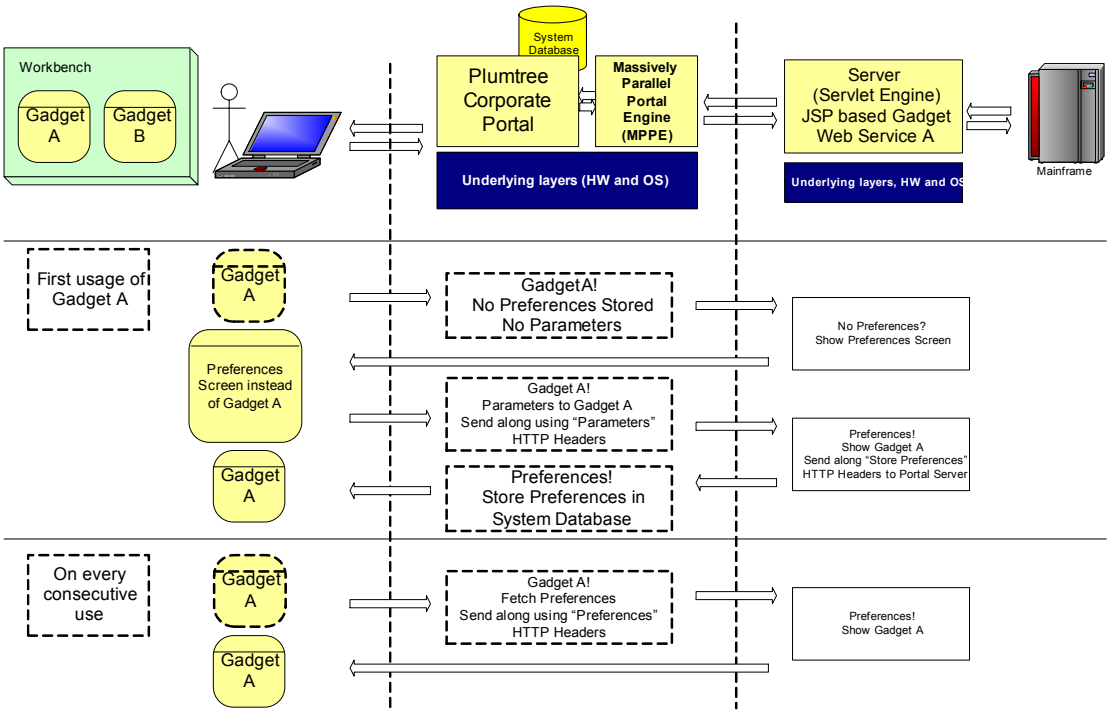


Figure 32 The Stages in Plumtree's Central Gadget Preferences Storage Handling

Plumtree lists several benefits that arise from this architecture. Amongst these are:

- **Scalability** – The parallel portal engine is able to handle multiple gadget servers for one gadget web service, distributing the load. On the other hand, multiple light gadget web services, requiring little server performance, can be hosted on a single server. This flexibility enables the administrator to tune performance by adjusting the number of physical servers that host the different gadget web services, adding physical servers to the heavy gadget web services.
- **Fault-tolerance** – If one gadget web service fails or "crashes", this affects only that gadget server. The physical server can also crash, merely disabling the portlets hosted on this server. The portal engine notices this, and "disables" the portlet server until the service recovers. If the gadget web service is deployed on multiple servers, the portal server will use these still working servers, and eventually include the failed server when the service is restored.
- **Security** – The gadget web services in effect isolates the enterprise application to which it integrates. The gadget web service decides which functionality of the enterprise application is exposed. By putting the enterprise application behind a firewall, and opening up for HTTP connections between the portal server and the gadget server, another layer of security is introduced, denying direct access from the portal server to the enterprise application.
- **Openness** – Plumtree's architecture is one of the most open portal systems around, since the developer may choose with which HTTP aware language portlet programming should be done. This is not judged as a "conformance" capability, as there isn't any portlet standard to conform to yet.
- **Technology Spanning** – Because of the loosely coupled system with HTTP communications, the gadget web services may run along with the technology to which it integrates, eliminating the need for bridges between the underlying

technologies. For example, a Microsoft Exchange integration gadget may be programmed on the Microsoft platform, using COM objects to communicate, and then expose the resulting service using HTTP. A Documentum integration gadget may be programmed in Java, using Java APIs to communicate, and again expose the resulting functionality using HTTP. The HTTP communications acts as the single bridge, defining a concise interface between the portal server and the portlets.

Wireless Device Server (WDS)

The WDS enables users to access specialized versions of their Workbenches through mobile devices. Supported devices include WAP phones, synchronized Palm PDAs and RIM Blackberry pagers. The server does not translate automatically between different markup languages. Plumtree reasons that [Plumtree C] "… the sheer variety of handheld devices combined with the dynamic and interactive nature of the content access by Plumtree portals makes this an unworkable strategy." Plumtree have instead focused on a platform where it easy to make gadgets useful in different settings. Gadget developers can specify which device types their gadgets are able to output information to, and the portal server will only display these portlets to the suitable devices.

Internet Device Server (IDS)

This is a server similar to the WDS. It is designed to accommodate for new products as they emerge, allowing the gadget writer to output information tailored to each device type. The IDS detects which device the user is logged on with.

Open Security Architecture with LDAP integration

The Plumtree portal claims a modular, open environment for inclusion of multiple authentication sources. Evidently, the portal server synchronizes its user database with external sources. Components called Authentication Source Providers is responsible for communicating with the authentication source. The portal is delivered with such providers for standard LDAP directories and Windows NT domains. The system also has integration for single-sign-on systems from Netegrity, Oblix, IBM and Securant.

Administrators can configure the synchronization feature to automatically give newly imported users a default profile. This profile may depend on their group memberships. The system can also read properties from the LDAP source and store this along with the profile. When needed, the server forwards this information to gadget web services as needed, thus enabling them to consider the LDAP properties when rendering the portlet.

Searching

All portal content is searchable using the Verity search engine, which is integrated into the portal. This search engine supports different search characteristics, including Boolean, proximity, fuzzy, linguistic stemming and advanced metadata. The search only includes documents that the user is allowed to see in the first place, thereby preventing that any access constraints are breached.

7.6.2 Framework concepts and features

The Plumtree's special two-stage rendering approach makes it difficult to talk of a single set of framework features. Plumtree portlets can be programmed in any "HTTP

aware language", thus including any programming language whose platform also includes a web server capable of CGI[47] scripting. This is in addition to programming environments made especially for the web, which includes HTTP aware constructs natively (for example ASP, JSP/Java Servlets and PHP)

Nevertheless, Plumtree have developed gadget development frameworks for some popular programming environments [Plumtree C]. These code libraries features functions that act as an API for accessing and setting preferences and administrative information that the Plumtree portal server has communicated to the gadget using HTTP headers and cookies. This relieves the programmer from the burdensome task of parsing these special HTTP headers and cookies, enabling him to focus on the programming of the gadget application logic.

Gadget Web Services are the portlets of the Plumtree portal system. Reviewed in the previous section, these will not be discussed further.

Enterprise Class Gadget Suites

An enterprise class gadget suite is a suite of portlets made for a specific enterprise application. Several enterprise applications are installed without much customization. For such systems, a gadget suite is made, featuring a set of predefined gadget web services, simplifying and accelerating deployment. In [Plumtree B] Plumtree states "This year, Plumtree is releasing Enterprise Class Gadget Suites for applications such as Microsoft Exchange, Documentum, Lotus Notes and Cognos e-Applications."

Enterprise Class Gadget Frameworks

A gadget framework is a separate framework made for a specific enterprise application. Instead of having a static set of predefined portlets, a gadget framework exposes application objects and functions from the underlying enterprise application. With help from a graphical "wizard", developers may assemble their own gadget web services in an efficient manner. The developer picks the objects he wishes to make a gadget web service from, and applies a set of framework functions to these objects to make them portal-ready. The developer may transform raw data sets to be presented in as portlets. Forms querying the user for arguments may be created.

The frameworks enables the developer to make forms to capture user and administrator preferences, and have functionality to store and retrieve those preferences. This is done using the HTTP header exchanges described in the previous section, but the burden of handling this exchange is offloaded from the developer, as the framework handles this.

The Plumtree portal features administrator-configured caching of portal content. In addition, an enterprise class gadget framework features another layer of caching on the portlet level. When the user reloads a page, or does a drill-down or similar activities, requiring the re-rendering of the portlet, the data is fetched from the cache instead of burdening the enterprise application with a new query.

[47] CGI – Common Gateway Interface. This is a simple protocol for how a web server communicates with external programs. HTTP headers, server variables and settings and similar properties are communicated to the invoked program using *environment variables* (variables that are available within the run-time of a program's execution). The output from the program is supplied back to the requesting browser.

In [Plumtree C], Plumtree states, “Plumtree is developing or has released Enterprise Class Gadget Frameworks for systems such as SAP R/3, PeopleSoft 8 and Siebel Sales.”

Plumtree is also developing a similar framework for developing stand-alone gadget web services, not requiring an underlying enterprise application. These portlets would contain their own application logic and data storage.

Multiple Workbenches and Communities

A user may define multiple “MyPages”, which are user defined and customizable workbenches.
Plumtree in addition have a concept of *communities*. Communities may be created by both the administrator of the portal server, and by special users who have the ability to approve portal content. A user may be enrolled in several communities. The community leader may enroll him, with a mandatory membership, or he may choose himself to enroll into communities to which he is eligible. The enrollment in a community gives the user a set of extra workbenches, which in effect is shared between the community members. The administrator may choose to display the extra workbench as a tab, while the employee-defined communities will usually list beneath a single tab. The workbenches include up to three columns of portlets.

Look and Feel

The user may choose between 18 different color schemes. The use of themes, where the entire graphical display of the workbench is changed, is apparently not supported.

Internationalization and localization

The Plumtree portal is already available in eight languages. All text strings are separated from the user interfaces and stored as separate XML files. This includes all help files. The character set used for internal representation is UNICODE, thereby being able to represent most any known language’s character sets. By choosing a locale, the user implicitly also selects a character set for his client. The portal server then converts the internal UNICODE representation to this character set before transmitting the content to the browser. The locale also sets other cultural conventions such as date, time and number formatting. Still, no information about how to program internationalized gadgets was found.

Invitations

The portal has mechanisms to enroll users into the portal with specific security profiles and ready-made homepages and workbench sets. This is specifically designed for B2B and B2C settings. The administrator may generate special URLs that generate an account instantly if followed. This feature enables viral marketing of the portal to target audiences, for example by sending out an email to the company’s customers, embedding a generated invitation link.

Hosted Gadgets

These gadget web services are hosted separately from the customer’s Plumtree installations. It may serve multiple customers, and may also allow customers to share services.

Network Search
Using this feature, a search may span multiple Plumtree installations across the Internet or corporate intranet. In addition, the search may span other searchable indexes; external internet and private intranet search engines. Plumtree has developed a platform-independent system for developing search services using the Simple Object Access Protocol (SOAP).

Syndicated Gadgets
This is a mechanism for delivering content to other portals or to systems that are not portals. This syndication feature enables administrators to output content either as XML or HTML on a scheduled basis. This enables easy exchange of content between for example business partners or customers.

User Profile
The portal server keeps a profile for each user. The profile includes which communities the user are a member of, which MyPages he have created and the layout of these, and the preferences for each portlet. These preferences are communicated to the Gadget Web Services when the user requests rendering of the workbench containing the portlet. This process is described under "Architectural Features".

Preferences
Plumtree supports different layers of preferences. Plumtree states "In fact, Plumtree supports five layers of preferences" [Plumtree C], but does not define these five layers in detail. Apparently, some of these are set by the portal administrator to define such properties as which physical server the enterprise application resides and credentials for how to connect to this resource. Other preferences are from the user, tailoring his specific view of the gadget to suit his needs. The last category mentioned are the community preferences, which are specific for community gadgets. These are not user configurable, but are instead configured for the whole community as a whole.

7.6.3 Software Quality Model References

Interoperability
Given the two-tier rendering approach, the Plumtree system may interoperate with every HTTP capable system. This eliminates the need for technology bridges, and is a considerable plus for the Plumtree portal. See also "Conformance".

In addition, Plumtree have developed and are developing several Enterprise Class Gadget Suites, which are complete sets of portlets, ready to be installed and used. Plumtree releases such suites for Microsoft Exchange, Documentum, Lotus Notes and Cognos e-Applications. Enterprise Class Gadget Frameworks are frameworks in which developers may easily create new portlets for specific enterprise applications. Plumtree have released a framework for SAP R/3, and are currently developing frameworks for PeopleSoft 8 and Siebel Sales.

Security
The Plumtree system states that due to the Internet Architecture, which introduces a distinction between the portal server and the servers running the portlet, the system is

more secure. This system introduces another layer of protection, as the portlet servers isolate the enterprise application from the portal server.

Fault tolerance / Recoverability / Availability

If one portlet crashes, only the portlet server on which it is residing is brought down. The portal server locks this server out, and uses any remaining servers with the same portlet. If there are no available servers running the same portlet, then the portlet is not shown to the users. When the server comes up again, the portal server starts to use it again. However, the portal server itself seems to be a single point of failure.

Adaptability

The Plumtree Wireless Device Server (WDS) and Internet Device Server (IDS) enables users to access specialized versions of their workbenches. The portlet developer must tailor the output from the portlets according to which device that is connected. The WDS currently only supports WAP phones, synchronized Palm PDAs and RIM Blackberry pagers. However, the IDS is designed to accommodate for new products as they emerge.

The Plumtree system is available in eight languages. All language specific strings are stored separately in XML files, including help files. Internally, UNICODE is user for character representation. Based on the browser's locale setting, the output is converted to an appropriate character set. The browser setting also decides other localization parameters such as date, time and number formatting.

Conformance

The portlet server is written in some platform native language, and is only available for the Microsoft and Solaris platforms. However, the portlets may be written in multiple standard languages; Plumtree supplies Gadget Development Kits for Active Server Pages, JavaServer Pages, Java Servlets, and Perl.

Scalability

The administrator chooses whether a portlet should run on a shared portlet server, alone on a physical server, or assign several physical servers to one portlet. If one portlet is used heavily, the Plumtree system lets the administrator easily add resources to the portlet, either by putting it one a dedicated server, or if that is not enough, assign multiple physical servers to it. However, again the portlet server itself seems to have the possibility of becoming a bottleneck.

8 Architecture and Architectural Features

This chapter summarizes and discusses the architecture and architectural features obtained from the preceding analyses.

8.1 High-Level Architecture

The highest-level architecture of all of the analyzed systems is similar to the one shown in Figure 33.

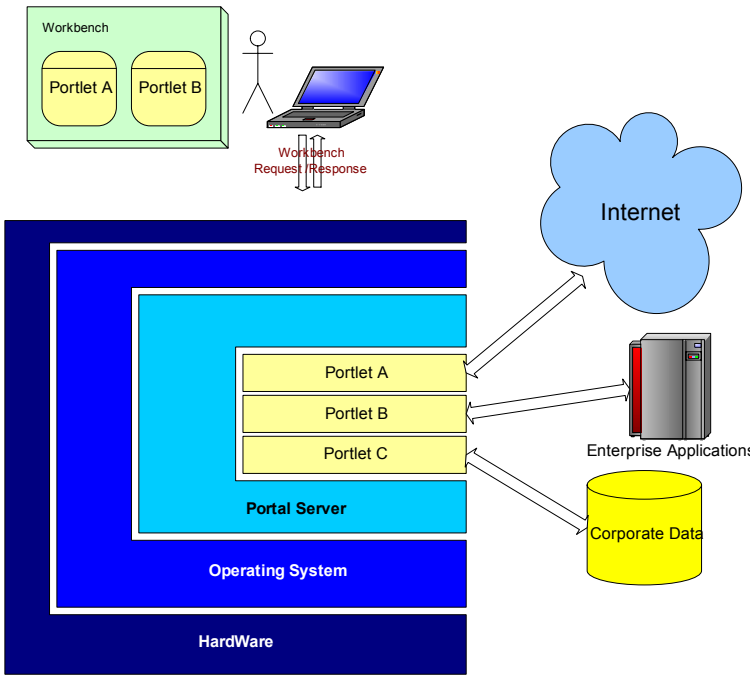


Figure 33 High-level Architectural Overview of Portal Systems

8.1.1 Portal System as Enterprise Operating System

As can be seen from Figure 33, the portal is providing a system in which portlets may execute. This concept resembles, to a considerable degree, a graphical operating system and the programs that are installed and is runs within it. The portal system represents the operating system, providing services to the programs running, some of the most important being the user and group management facilities, and user interface and window management systems.

The responsibilities of a portal system is remarkably similar to the responsibilities of an operating system, some are mentioned in the following list:

- User interfacing
- User management
- Data source management
- Running and scheduling of programs, multi tasking
- Inter-process communication

The portlets are representing the programs running within the operating system. The portlets can execute independent of each other, and the user may interact with each specific portlet without the other portlets being interfered with.

By enabling the portal to have portlet applications easily installed and managed by the portal administrator or administrators, the portal system in effect becomes an enterprise wide operating system.

8.1.2 Architectural Standpoint: Webpage Assembly or Operating System

While analyzing the different systems, an impression of two different architectural standpoints arises. On one hand, it can seem like some vendors have understood the portal paradigm as an interesting new way of assembling a web page, customizable by the user. The portal concept is merely an evolvement from the vendors' existing architecture. For example, the BEA portal solution use BEA's existing Webflow system in conjunction with an extensive recursive inclusion of different JSP files to assemble a set of customized views. The system does not provide for any user specified pages. To a certain degree, this is also true for the Jetspeed system, where Turbine provides for Pages, Actions and similar constructs, and Jetspeed extends this approach to a customizable workbench, using the same mechanisms[48].

In contrast, other vendors seem to view the portal systems as an entirely new concept, demanding a new environment. The systems from by Plumtree, Epicentric, IBM and Oracle resemble new operating systems. The portlets are a new entity with its own "life". The portal paradigm is not only about the assembling of a view, but is addition a new programming paradigm, where new thinking must be employed. The system from IBM has taken this thinking to the fullest extent, employing a new, stand-alone API on which the portlets are programmed. This API can potentially be implemented by other vendors, enabling the portlets to run on other portal servers than IBM's.

8.1.3 One or Two-tier Rendering

Since multiple portlets exist as rendered windows on single web page, all of the portlets and their containing windows must actually be rendered for each action the user does. There are two different approaches to this operation. It is important to note that this discussion only relates to how the portlets themselves are executed, and not to whether a portlet must consult one or more external tiers to fetch the data used to generate the content in the portlet.

One-tier rendering

In these systems, one server is both working as the portal engine and the environment in which the portlets are executing. The control is alternated back and forth between the portal engine and the portlets on the workbenches, to some degree resembling a multitasking operating system. Passing of the parameters from the user and the portal engine, and the return of output-markup from the portlet, is facilitated using method invocations directly on the portlets through the portlet interface. This method is shown in Figure 34, and is used by all vendors except Plumtree and Oracle.

[48] However, the history behind Jetspeed and Turbine are actually the other way around, where Turbine was "extracted" out from Jetspeed.

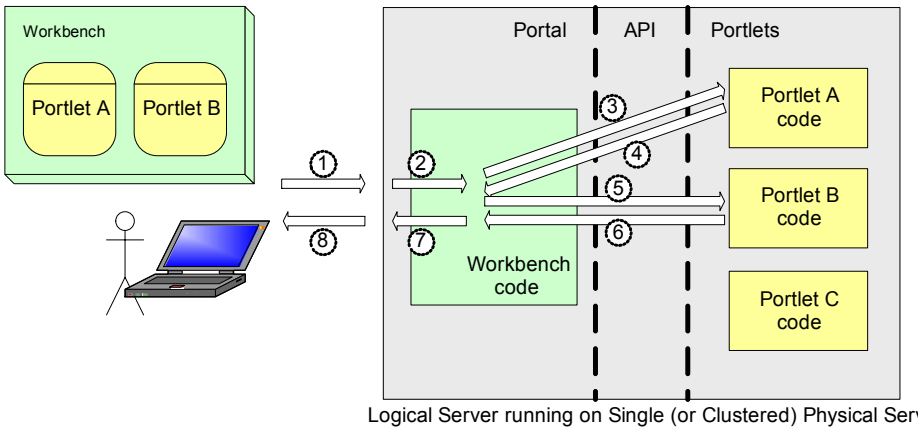


Figure 34 One-tier sequential rendering

Two-tier rendering

This approach relies on using multiple servers. The logical portal server functions as a hub, invoking the portlets' code residing on other logical servers. These portlet servers are usually residing on different physical servers as well. Passing of parameters to the portlets and return of the output from the portlet is facilitated by some protocol over a TCP connection. The Plumtree system uses a set of HTTP headers to pass control information, and the ordinary HTTP protocol to pass data. The Oracle system uses SOAP over HTTP for both control and data. This method is shown in Figure 35.

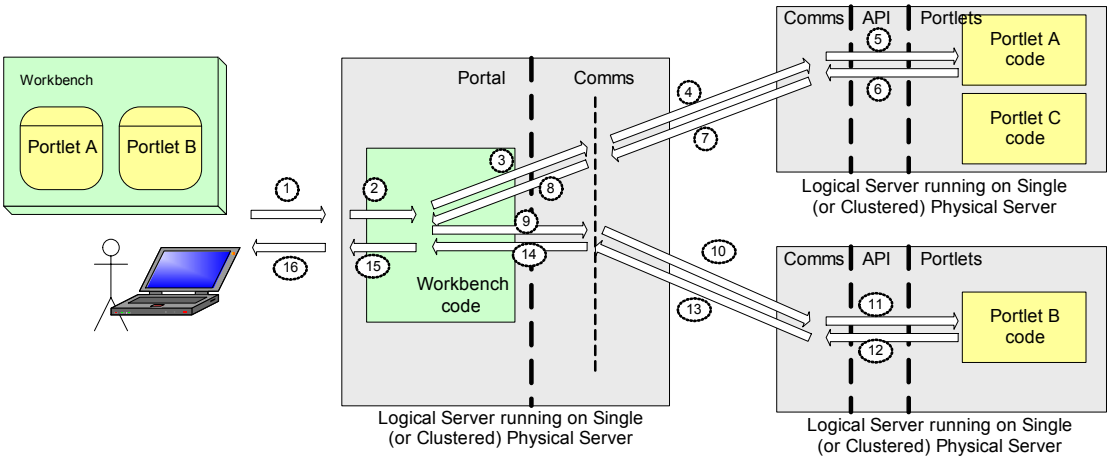


Figure 35 Two-tier sequential rendering

An enormous advantage to the two-tier approach is that the portlets does not necessarily need to be programmed in the same language as the portal server. Since the communication occurs over standard network connections, using standard HTTP, any programming language on any system can be employed to write portlets. Plumtree supplies Gadget Development Kits for Microsoft's Active Server Pages, Java ServerPages and the programming language Perl. Oracle has made a Portlet Development Kits for Java and for their proprietary PL/SQL database language. In addition, they supply beta versions of such PDKs for web based portlets that are fetching information from simple web pages, and for interfacing to the emerging Web Services.

Since portlets can be programmed in any language, technology bridges are eliminated. The portlet can be programmed in the same language, on the same platform, as the underlying resource it will access.

The system is scalable in the way that an administrator can place the individual portlets on different physical servers. Some portlets may be hosted together on one physical server, while other, more resource-intensive portlets can be hosted alone on one physical server, or in Plumtree's system, on multiple physical servers. However, there is a potential bottleneck with the saturation of the portal server itself.

Faults are effectively isolated since one portlet cannot possibly the whole server. The Plumtree server stops accessing the portlet if it crashes or times out several times, and instead starts polling regularly to check whether is has come up and is acting normally again. Nevertheless, the portal server itself is a single point of failure.

Plumtree also argues for a heightened security with this model. As the portal server does not access the underlying sources directly, but have to go through the portlet server, one extra layer of security is introduced.

8.1.4 Parallelizable Portlet Rendering

All portlets on a workbench must at some point supply their content to the user. There are two different approaches to how the invocation of the multiple portlets sitting on one page should be done. The first is the obvious one of sequential rendering, where each portlet is executed in turn, their output being collected and sent to the user's browser. However, there is one big drawback to this. Many portlets' main task is to fetch information from underlying databases and systems, thus each portlet may experience some amount of input/output ("I/O") latency and wait-time in addition to the actual CPU-time it takes to execute the rendering. Since each portlet is executed in sequence, the total amount of rendering time ends up being the sum of these sub operations.
In this light, the Plumtree solution is employing an interesting and actually rather obvious solution of parallelizing these tasks. The Oracle documentation also talked of "parallel" at some point, however it was difficult to deduce whether this was at the same level as Plumtree. This method is shown in Figure 36.

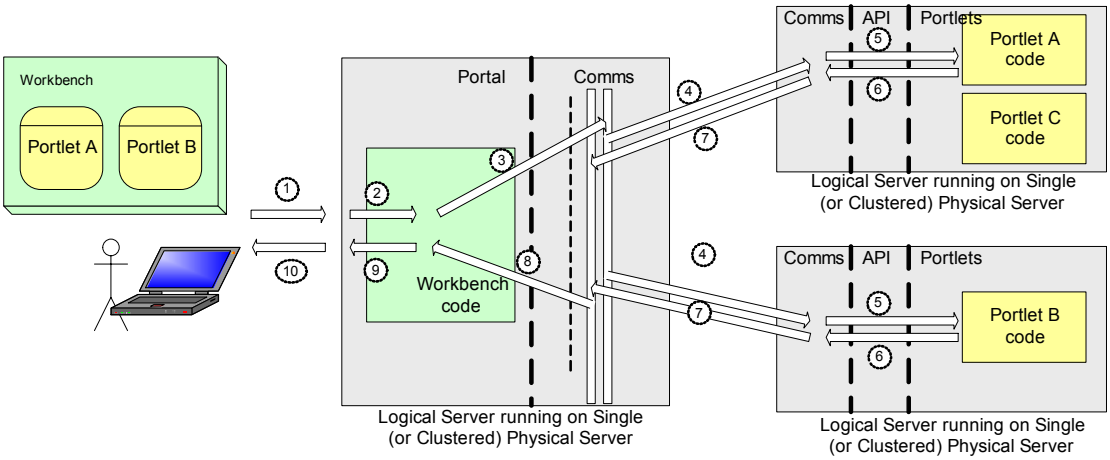


Figure 36 Parallel two-tier rendering, as used in the Plumtree Parallel Portal Engine. The two portlets, Portlet A and Portlet B, are executed simultaneously.

It is worth to notice that the two solutions employing this method are the same two solutions that use the two-tier approach to rendering, explained in the preceding section.

8.1.5 Portlet Caching

By enabling a portlet to state that its content has not changed since the last execution, one could relieve the server from executing portlets at all by caching and reusing the content from each invocation. This is what most portals are doing.

A portlet may be cached at multiple levels. Some portlets are showing the same content to all users of the portal, other portlets' content are coupled to a group of users, while other still are tied to each individual user. The caching mechanism enables the portlets to control at which level their content could be cached, and to control for how long the caching should occur.

More on caching from the developers point of view is found in Chapter 11 – "Framework Features".

8.2 Platform and Language

8.2.1 Dominant Language and Platform: Java and J2EE

All analyzed portal frameworks except one is using Java as their implementation platform. More specifically, the Java 2 Enterprise Edition is employed to some degree, from full utilization of all of J2EE's features, to a dependency only on the Servlets part of the J2EE specifications. The technology stack for these J2EE-based systems is similar to the one shown in Figure 37.

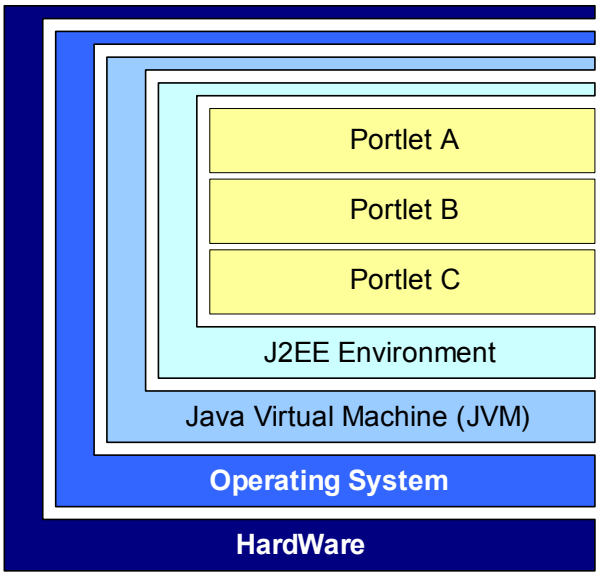


Figure 37 Technology stack for J2EE based portals

Plumtree have made a portal server based on some compiled language; it is only available for the Microsoft and Solaris platform. However, Plumtree also provides a Gadget Development Kit (GDK) for the Java platform (Gadgets are Plumtree's name for portlets).

One major benefit resulting from the use of the Java platform is platform independency. Java Virtual Machines exists for most operating systems. Several OSes have multiple JVM implementations, several of which are free. Multiple Open Source Java Servlet Containers are available[49], in addition to a fully-fledged Java 2 Enterprise Edition environment[50]. This makes the platform very flexible, and a systems administrators and developers may decide whether they would like to use commercially developed and supported operating systems, virtual machines and J2EE platforms, or go for open source, free solutions. In addition, combinations of these are possible, as an example, a corporation could decide to use the open source operating system Linux, the free Java Virtual Machine from Sun, Oracle's database and IBM's J2EE server, on which the Epicentric Portal solution could run. Commercially developed J2EE environments are often scalable, thus being able to support a very large number of users. An overview over the J2EE platform is given in Appendix B.

8.2.2 Vendor Lock-In

One of the leading reasons for the J2EE standardization effort is to provide corporations with flexibility to choose between different systems when developing multi-tier applications and enterprise services.

However, it seems like several vendors use their portal product to leverage their own application server. This is indeed also noted by Vizard [2002] in an article in InfoWorld. Where a portal solution could have been developed on any conformant J2EE structure, vendors with an existing application server platform have instead decided to require the use of their own application server. Oracle goes so far as to require both the Oracle application server and database solution to use the Oracle Portal. In addition, choosing a vendor's application server opens the way to using a whole host of different add-ons to this server, each plugging especially easy into the vendor's own framework, and poorly into any other framework. In this way, every vendor tries to lock their customers tighter and tighter into the vendor's full product suites.

In many ways, it can seem like vendors actually tries to find places where the standard does not define any particular behavior. Such holes are exploited mercilessly, developing incompatible solutions, luring the customer to buy additional software from the same vendor.

This attitude can be seen to some degree with the vendors having an existing application server product: BEA, IBM and Oracle. Diametrically, Epicentric's portal system is also built upon the J2EE architecture, but this vendor does not have their own application server, and thus boasts of the large number of different application servers the portal can run on.

With the standardization of a Portlet API and web services portlets, this will have to change. More on the standardization efforts and their impact is found in Chapter 12 – "Portal Framework Evolvement".

[49] Apache Jakarta Tomcat: http://jakarta.apache.org/tomcat/index.html, Jetty: http://jetty.mortbay.com/jetty/index.html, GNU Paperclips: http://savannah.gnu.org/projects/paperclips
[50] JBoss: http://www.jboss.org

8.2.3 Integration with J2EE

For the systems that are written in Java, the degree of interaction between the J2EE architecture and the platform system varies considerably. There are several areas where this is seen. First, while most Java based portal systems require the entire J2EE platform to run, at least one other only requires a servlet container to run[51]. Next, some systems use the user authentication system already defined in the J2EE standards, while others have chosen to use application level authentication. Lastly, the degree to which Servlets and JSPs are employed in the portal architecture varies greatly.

J2EE vs. Servlets

BEA Portal requires the BEA J2EE application server. The system uses several web applications in addition to multiple Java Enterprise Beans to run their portal solution, thus requiring the full J2EE specification. IBM just changed their portal server's API to require that each portlet also were a servlet. This means that each portlet application is defined as one web application, which again implies that a full portal solution would require communication between multiple web applications.

Several of the developers on the mailing lists of open source projects feel that the full J2EE architecture with its enterprise applications is "bloated" and requires too much overhead. Turbine and Jetspeed does not require this, but runs on the Servlet Container alone. The IBM portal that just got discontinued in favor for the new version 4.1 also ran on the servlet container alone.

User Management

BEA seems very dedicated to the J2EE environment, and uses the architecture's user authentication procedures. This seems to be the only one, as the other implementations employ application level user authentication mechanism. The Turbine/Jetspeed teams reason that container-managed security can not easily be managed from within the application. The idea with container-managed security is that some container specific method for authentication is employed, for example a database or LDAP backed mechanisms. There is however no defined API relating to how to control the authentication mechanism's store of users, nor is there as standard set of features that should be embedded within the store, for example group or department membership. This limits the applications flexibility relating the dynamic editing of existing accounts, and creation of new accounts, for example by "invitations" as in the Plumtree solution. In addition, the lack of a standard group membership solution severely restricts the creation of expressive access control systems.

Servlets and JSPs vs. Other APIs

The BEA solution exclusively uses recursive invocations of JSPs to render their portlets, and does in addition use a set of Enterprise Java Beans. The API that IBM used in their 1.2 and 2.1 version of the WebSphere Portal was a stand-alone API developed in conjunction with Apache Jetspeed. However, in their version 4.1, IBM decided to rely on the Servlet API by making the Portlet interface directly extend the Servlet interface. Both IBM and BEA's solution place an extensive use of the hosting J2EE environment.

[51] The servlet container is defined as a sub component of the J2EE standard, confer Appendix B

In comparison, the other portal systems provide their own stand-alone API, not related to the Java 2 Enterprise Edition. The ones using a servlet container apparently just were programmed on the servlet platform by way of convenience, as the servlet framework is considered by most Java enthusiasts as a mature and developed technology. IBM's earlier portlet API had similarities with the Servlet API, but this was more of a cosmetic similarity to help servlet programmers migrate to the new portlet paradigm.

The portal systems, whose APIs do not depend on the any of the J2EE specifications, could potentially be developed as stand-alone servers, without the requirement of a servlet container at all. It is hard to decide how big benefit this really is, though.

8.3 Rendering

The rendering of the portal page is another area where different approaches have been chosen by the different vendors. Both the laying out of the portlets, how the portlet is constructed, and how the windows around the portlet's content is rendered is done in multiple ways.

8.3.1 Layouts

The portlet server must use some rendering mechanism to include the content from multiple portlets. On the HTML based output systems, HTML tables are being utilized to divide the workbench area into the correct pieces. A common way to do this is by dividing the screen into three columns, with the two side columns being quite narrow, and the one in the middle much wider. The portlets are stacked on top of each other inside these columns, and can thus be as tall as they like, while the width is fixed.

However, there is no reason why the portal developer could not control the layout system too. This is tended to in the BEA and Apache Jetspeed systems. On both these systems, the developer have extended control of how the system lays out the portlets.

In the BEA system, the developer configures the different layout within the system configuration files. The JSP file responsible for the layout contains markers as to where the different portlets should be inserted. In addition, the developer tells the system how many columns there are in total, and whether the portlets within each column should be stacked on top of each other in a vertical fashion, or if they should be placed at the side of each other, forming a horizontal column. This information makes the system able to present a configuration screen to the users, letting them place the portlets of their choice into the different columns.

The Jetspeed system uses a control called PortletController to render a PortletSet. The controller lays out the set of portlets within the portlet set in a specific fashion. The system is shipped with a whole set of controllers, and the developer may develop new controllers that lay out the portlets in other patterns. In addition, the user is given considerable control over the layout of his workbenches. On a workbench, the user may choose to add a new set of workbenches, selectable by tabs. In this way, a hierarchy of workbenches may be constructed.

8.3.2 Portlet Modes and Portlet Rendering

The portlets are usually contained within windows that must be rendered by some mechanism. In addition, portlets may be displayed in several modes, usually activated by clicking some button on the containing window frame. The different action buttons on the window frame must be handled by some mechanism.

In Jetspeed, the portlet window is handled by an object called a PortletControl. This object contains a Portlet, and is also itself a Portlet. The portlet control both generates the output of the windowing markup, and acts as the window manager for the containing portlet. It controls in which mode the portlet should be shown, being it minimized, maximized or invisible, and handles the actions that are done on the window. All other methods are delegated to the contained portlet or portlet set.

In most other systems, the portal engine handles the windowing and modes switching as base functionality. The window markup generation is always changeable to enable themes-support.

The BEA system have a very elaborate portlet rendering approach. The developers specifies a set of jsp files, including the titlebar, banner, header, portlet and footer. In addition, if the portlet supports the help and edit modes, a jsp file for those modes must be supplied as well. Usually, the titlebar file is selected to be the system default, as this supports all the necessary features like mode switching.

The number of modes natively supported by each portal system differs. The combined list of modes are as follows:

- *Minimized*: Only the portlets title bar is showing.
- *Maximized*: The portlet is using the entire workbench.
- *Edit*: The portlet is in a defined edit mode, allowing the data to be changed by the user.
- *Floating* or *detached*: The portlet is detached from the workbench, and fills the entire area of a separate browser window. The window markup is not drawn, as the browser functions as the window.
- *Help*: several systems natively support a help button on the window's frame, opening a help screen in a new browser.

The edit mode and floating mode is the least supported modes. The idea with the edit mode is that it helps the developer distinguish between when the user is supposed only to view data, and when the user is allowed to edit the data. For example, the IBM system only allows the customization data to be changed if the portlet is in edit mode, raising an exception if this is attempted in other modes. The BEA system lets the developer assign different JSP files to the different modes.

8.4 Configuration and User Management

Most portal systems have some file based configuration details. However, web-based interfaces are the norm for configuring most parts the systems. The reason for this approach is that graphical user interface is much easier for most people to handle than text editing of properties-files or XML files, especially when the administrator is new to the technology.

Configuration details include user and group management, default workbench layouts and access control list editing for the different elements of the portal system and other administrative tasks. For further discussion of this topic, confer Chapters 9.5 – "Administration and Delegation" and 10.4 – "Portal Creation", Internal vs. External User Management

The user, group and access control management of a portal system is a complex affair. Authentication only refers to the accept or denial of a user's credentials. This can be handled by any system that can store a username associated with a password. However, portal systems often rely on group memberships and comprehensive access descriptions for each user. These details are part of the user's *profile*. Connecting to, for example, an existing LDAP source usually only supports the authentication, while the group memberships and access control lists must be handled by some internal means.

Plumtree handles this by synchronizing its internal user database with the external sources. This is done using a set of components called Authentication Source Providers, which are configured to regularly update the internal user system with newly added users from the external source. These new users can be given a default profile upon import.

However, the Oracle Portal uses the Oracle Internet Directory as its repository for user and group definitions. The user and group structure may be administered through the built-in administration interface embedded in the portal, or using the administration tools and APIs delivered with the directory system.

8.5 Preinstalled Modules

Several systems are preconfigured with a set of modules. Popular are the inclusion of some content management system, essential to any portal system. In addition, system taxonomy and searching functionality is often included.

Epicentric uses an external system for its searching facilities, the Inktomi Enterprise Search engine. This is installed with the portal. Plumtree does also use an external system for searching, the Verity search engine.

9 Framework Concepts

This chapter summarizes and discusses the framework concepts obtained from the preceding analyses. Several framework concepts are already mentioned in Chapter 5.3 – "Functional Components of the Corporate Portal". The reason for including some of these concepts again in this chapter is that this is a more low-level view of how a concept is used in a portal development situation, and how the concept is implemented in the portal system.

9.1 Customization, Preferences and Personalization

Customization, preferences and personalization refers to very similar concepts, but refers to different levels of implicit or explicit control. Customizations and preferences are explicit configurations done by the user, like selecting which portlets that should be shown on a workbench and how the discussion forums should display their posts. Personalization refers to implicit adaptations made by the framework, based on who the user are and configurations done by the administrators.

Customization and Preferences

When a user uses some configuration portlet, and for example adjusts the way his workbenches appear, it is referred to as customization. Customizable elements in the analyzed systems include:

- Workbenches
  - Creation and editing of personal and group workbenches
  - How the portlets should be arranged; the layout
  - Which portlets should be visible and in which order should they be shown
  - Joining of group workbenches
- Theme / Skin
- Localization
  - Language
  - Date, Time, Numbers and Currencies

In addition, several portlets also have user configurable elements, for example, from which field and how many news items the news portlet should display. This is often called *preferences*. However, some portlets needs to have some data to work properly, for example, an email client needs the server, username and password to be used at all. This is usually called the *configuration* of the portlet. More on this is provided in Chapter 10.7 – "State, Persistence, Preferences and Configuration".

Personalization

The personalization refers to the concept of tailoring the user experience in accordance with who the user is. As the BEA portal system analyzed included the most elaborate personalization system, this will be used as a reference.

The BEA system includes a rich personalization module. This module uses two concepts called *characteristics* and *entitlements*. A characteristic is an administrator specified attribute, of which each user has to be assigned a value. For example, the user's "SkillLevel" must be set to one of "Low", "Medium" or "High". An

entitlement is a property that is automatically assigned if the user meets a set of conditions set by the administrator. Attributes that conditions may build upon include the user's characteristics, groups that the user is assigned to, the IP address the user connects from, and the date and time. Each of these attribute can be required to be an exact value or within some range of values. The conditions may be logically or'ed or and'ed together, and can thus form a complex set of requirements.

A system may contain a large set of different entitlements, each with its own conditions. A user is often assigned multiple entitlements. An entitlement is created with the entitlement editor. An administrator may tick off the different types of conditions available, and more closely specify when the condition applies.

Every portlet in the system can now ask whether the active user has the different entitlements. This forms a powerful tool by which to tailor the appearance of the portal's pages to each individual user. As a simple example, the system may show a user who is entitled "ExperiencedJavaDeveloper" a new article about some advanced Java topic, and users entitled "HumanResourcesEastCoast" must fill out survey about their subjective impression about employee satisfaction. If a human resources person from the east coast happens to be an experienced java developer, he or she will both see the article and the survey.

9.2 Security

9.2.1 Access Control

Most portals solve the access control problems by using a set of elements, namely *users* and *groups*, and *permissions* and *roles*. As these concepts are elaborated on in Chapter 6 – "Basic Portal Functionality", they will not be reiterated here.

ACLs are particularly used in the Oracle portal to delegate administration, and to control the user's access to different portlets and other features.

9.2.2 Single Sign-on

The Single Sign-on concept is also elaborated on in Chapter 6 – "Basic Portal Functionality", and will not be further discussed. However, an interesting, yet simple service provided with the IBM portal system is the *credentials vault*. This is a system facilitating single sign-on by keeping a user's credentials for different systems in a protected and encrypted store. The store can both be a system default, or interface towards already existing single sign-on system that already have a credentials store in place. The credentials vault may be used in two ways. The first, most straightforward, is to use the vault as a bank, requesting a specific user's credentials for a specific resource, and subsequently establish and programmatically authenticate the connection to the resource. The second, more secure method, is to ask the credentials vault system to establish and return an authenticated connection for the specific user to the specific resource. This latter approach is the preferred one, as the actual credentials (for example a username and a password) never are exposed to a potential malicious portlet.

9.3 Administration and Delegation

The portal administration consists of many tasks, including creation of the portal, managing users and groups, managing access to the different objects within the portal and creating workbenches for specific groups of users. Most portal system has a distinct focus on administration and the delegation of administrative tasks to multiple individuals. The reasoning behind such functionality is that today's large corporations are so large and spread-out that one administrator cannot handle all the necessary tasks, and that locally situated administrator have better local control. Several portal systems emphasize their ability to create multiple portals on the fly, within one server. These new portals can usually be administered by a dedicated administrator.

BEA has a distinct set of administration levels. These are system, portal and group administrator. These levels go along with the distinct set of different portal levels. The system administrator controls the entire enterprise application, thus including all portal web applications contained within. This level controls the creation of portal web applications, and all levels beneath. A portal web application is administrated by a portal administrator. Within one portal, one can create group portals. These are subsets of the containing portal, and are controlled by group administrators.

BEA uses explicit delegation. A system administrator can create several portal and group administrators. He can then grant the administration of one portal web application to a portal administrator, and at the same time, he can choose whether this administrator may delegate the same right further on. This enables the system administrator to select a few portal administrators, and enable these to further delegate the group administration tasks. Alternatively, the system administrator can decide that it is best if he controls the entire delegation, having full control. He would then take away the delegate right from the portal and group administrators.

All administration of the BEA portal, except the creation of a new portal web application, is done using the Portal Administration Tools. This menu-like interface gives access to user and group management facilities, as well as portal management facilities.

Epicentric also have the notion of multiple portals within a server, and the server administrator can grant administration rights to the portals' owner or owners. However, he may restrict the administrative capabilities using permissions. By giving subsets of portal administration rights to several people, *collaborative site management* is achieved. All administration of the Epicentric portal is done using the *Site Console*. With this console, the administrator may manage all aspects of the portal, including the workbenches, portlets and styles. A central *Server Console* gives access to all portal sites. This is located within the *Portal Command Center*. An interesting detail of Epicentric's taxonomy is that all parts of it are protected by permissions. Thus, the administrator may delegate the population and structure of each part of the taxonomy to different persons within the corporation.

Oracle's portal system is protected throughout by its ACL system. This includes pages (workbenches), styles, items and portlets. Each element has its own access control list, and privileges for accessing, customizing and modifying the different elements are granted to users and groups. In addition, an administrator may delegate the management of the access control list for each object to the object's owner. An

administrator may also grant global privileges to sub-administrators on all objects of a given type.

9.4 Modules and Portlet Applications

Modules or Portlet Applications are a set of portlets that together form a functional unit. Much as an application is installed on an operation system, a portlet application is installed into the portal. The portlets contained within the portlet application is configured by the administrator, and is thus made available to all of the portal's users.

The old IBM system, using the new Jetspeed API, had a concept of a *portlet application archive*, where an administrator could load and unload such packaged "par"-files dynamically. The new IBM portal uses the underlying J2EE environment's web application concept to contain a portlet application, which places the burden of dynamical loading and unloading on this servlet container. Oracle has a concept of remotely hosted *Providers*, containing the actual portlet implementations. The URLs to the location of these providers are configured while the system is live. The server asks the provider which portlets it contains, and these are immediately made available to the users of the portal, possibly restricted by access rights set by the administrator.

9.5 Services

Some of the portal solutions analyzed provide for a concept of *services*, which are "by-name" locatable resources. A service is registered with the portal server using the portal's configuration files. The registration includes some id or name by which it may be located. The portlet programmer can fetch the service, or an object representing the service, by invoking some globally accessible method, for example getService(id). The id refers to the id by which the service registered. If the server does not have the requested service registered, it can raise an exception, or just return "null".

By encapsulating common portlet tasks, services in effect provide a standardized extension mechanism of the portal's built-in APIs.

Jetspeed, for example, by default provides several services in addition to the ones already provided by Turbine, Jetspeed's underlying framework. These includes a service to manage the groups and users, a service to help portal developers cache the portlets contents and services to access the roles management and ACL systems. IBM also provides the services concept, and includes an interesting service called the credentials vault, which was elaborated on in the security subchapter.

9.6 Multiple Workbenches

All portals support the concept of multiple workbenches, and all except BEA's system support the dynamic creation of new workbenches.

The users of a portal can most often create their own, personal workbenches. In addition, most systems provide for some concept of *group* or *community workbenches*. These are workbenches which the user either chooses to accept, or are just given without any explicit accept from the user.

9.7 Invitations

One feature of the Plumtree portal worth mentioning is the concept of *invitations*. The portal can generate special URLs that, when followed, will enroll a person to the portal's user database. The portal generates a default user profile for the newly enrolled user. This system is especially meant for B2B and B2C settings, where for example viral marketing of the portal is desired. An email can be sent out to all of the company's customers with such invitation links embedded.

9.8 Remote Portlets

Plumtree and Epicentric use a concept of remote portlets. As the Plumtree system relies on the two-tier rendering system, in actuality all portlets are remote. However, they can be also be hosted on another company's installation, and one portlet on a portlet server can thus support multiple customers.

Epicentric have started the Web Services User Interface [WSUI] initiative, a SOAP/XML approach to hosting portlets at remote sites. This system, in addition to the Web Services Remote Protocol [WSRP], will be explained more thoroughly in Chapter 12 – "Portal Framework Evolvement".

9.9 Internationalization

Most vendors boast that their systems are internationalized; Oracle claims that their portal is localized to 27 languages, and Epicentric mentions eight languages.

However, it is obvious that internationalization has not been the primary focus for any of the portal vendors. The focus, if any, is on language support, and all vendors except from Epicentric seems to have forgotten about other internationalization problems like character sets, text direction, dates, times, numbers and currencies. Epicentric mentions in one sentence that they support multiple character sets, dates and times.

This is strange, considering that the Java system and the HTML standard are internationalized to a remarkable degree, tending to all the issues mentioned. When both the vendors' documentation have very little focus on this issue, and the user abstraction objects analyzed does not have any attributes concerning localization, an impression of pure neglect arises.

9.10 Statistics and Business Metrics

The BEA system features a set of developer-centered statistics, built into the J2EE server. The system times each run of the JSPs and servlets in the system. Displayed numbers are minimum, maximum and mean run-times, in addition to the number of runs each resource have had. These are invaluable to a developer as an easy way to get a feel of the status of the server, in addition to information on where to focus tuning and optimizing efforts.

Epicentric have a concept of *business metrics*, referring to statistics tailored to measure the effectiveness of different portal sites. The system comes with a set of metrics, and the administrators may develop their own. The installed metrics include static numbers like registered users, pages and sites contained in the system. Dynamic metrics include users using each portlet, active users, page views and minimum and maximum number of concurrent users.

10 Framework Features

This chapter summarizes and discusses the framework features obtained from the preceding analyses. Since the amounts of low-level documentation available varied considerably from vendor to vendor, some features may have been missed.

10.1 User Abstraction Object

Several of the portal system analyzed features some kind of *user-object*. This is an abstraction of the logged on user, giving access to user specific properties and attributes. For example, the user object defined in the IBM API gives direct access to the user's full name, nick name, his user ID, the last login time and a set of predefined attributes, accessible via a getAttribute(name) method on the user object. Apache Turbine's user object is fetched from the omnipresent RunData object. Apache Jetspeed is built on Turbine, and this feature is thus inherited. The Turbine user object does also have accessors for such personal data, and in addition features temporary and permanent storage for user values. These storage features support *state handling*, elaborated on in Chapter 10.7 – "State, Persistence, Preferences and Configuration".

By making available such an object, the framework relieves the portlet developer from the burden of storing such common information. The user object provides a platform for enabling common data about the user to be shared by all portlets.

Several frameworks provide a non-authenticated user object if the user accessing the portal is unknown or not yet authenticated to the framework. This enables the portal to be used by both authenticated and non-authenticated users, as the developer can program one portlet that will be shown to both types of users. If some information requires authentication, the developer can have the portlet check whether the current user object is the special non-authenticated one, or, if the framework have provisions, ask the user object whether it represents an authenticated, valid user.

10.2 Multiple Markup Support and Client Capabilities

Several of the portal systems analyzed support multiple markup languages on different clients. There are several aspects to multi-client access. Multiple markup languages must be supported, for example WML for WAP mobile telephones and HTML for ordinary web browsers. However, both of these markup languages come in different versions, where the older versions usually support less features than the new ones. In addition, different browsers may support different features, or have known bugs with some of their features, as experienced for a long time with the Netscape and Internet Explorer fights.

10.2.1 Multiple Markup Languages Support

Plumtree, Apache Jetspeed and IBM's systems support multiple markup languages, specifically WML. These vendors require the portlet to handle each type of markup. There are two main issues with multiple devices and markups: screen size and markup expressiveness. The screen sizes vary considerably, from for example 80x40 pixels monochrome on a WAP telephone, to 1600x1200 pixels with "true color" on a computer screen. The WML markup for WAP devices is completely different, and supports fewer features, than HTML for ordinary web browsers.

Most supported is the online access of a mobile WAP telephone, supporting the WML markup language. In addition, the Plumtree portal support synchronization systems like the one from AvantGo. The synchronization scheme is directed towards static content, and interactive services are not supported.

An interesting approach to this challenge is to make a device-independent markup language, for example using XML, and transform this using some transformation mechanism, to the different markup languages supported by the different clients. However, given the large span of screen sizes, it would be very difficult to both fit everything into the small screens and at the same time make use of the large areas that web browsers on computers typically support. This would probably require the device-independent markup to include "if"-like constructs regarding the screen size, and device-independence is in effect not gained. In addition, this device-independent markup would have to be adjusted towards the lowest common denominator regarding the markup language's potential. This would lead to the situation where no system's full capabilities would be utilized. This has been discussed on the Jetspeed mailing lists in conjunction with the discussions about the new portlet API[52].

The systems supporting multiple markups lets the portlets know which client is hooked up, so that the portlet may format its output according to the markup supported by the client.

10.2.2 Client Capabilities

The IBM system takes this approach further, and makes available an object representing the client. This object can be queried for which markup the client supports, and if it supports multiple markups, the developer will get a list in decreasing order of preference. In addition, the developer may ask what capabilities the client supports, for example, whether the client supports JavaScript, frames, tables and so on.

10.3 Portal Creation

Before a portal solution can be used, the portal system must be installed, and the actual portal must be defined, created and configured. Installation is usually facilitated by some "wizard" approach, guiding the administrator through the necessary steps. However, the logical definition of a portal can be a daunting task, with the portal, workbenches, portlets, users, groups, access restrictions and so on needing to be defined. This subchapter will treat the creation of the actual portal, while Chapter 9.3 – "Administration and Delegation" treated the administration of the portal.

10.3.1 Multiple Portals within a Single Portal Server

Several of the portal systems analyzed can contain multiple portals within one server. The reasoning is that several large corporations would like to be able to dynamically create new portal initiatives on demand.

The different portals are accessible either as a path-extensions to the URL, for example http://company.com/portalA and http://company.com/portalB, or as a

[52] Santiago Gala's post about multidevice rendering: http://www.mail-archive.com/jetspeed@list.working-dogs.com/msg05108.html, Thomas Schaeck (IBM)'s follow-up: http://www.mail-archive.com/jetspeed@list.working-dogs.com/msg05149.html

separate internet address, for example http://portalA.company.com/ and http://portalB.company.com/. Epicentric calls the different portals for *portal sites*, and the collection of portal sites residing on one server, a *portal network*. Oracle refers to the portals as *hosted portals*, and thinks of the collection as an *ASP model*. The server administrator can delegate the administration of such portals to other users, making them *portal administrators* or *site administrators*.

10.3.2 Portal Creation

Two different methods of creating portals are employed, either using some graphical user interface, generally an administration interface within the portal itself, or with some direct file handling, copying files or using a text editor to edit configuration files.

Visual Creation

The different vendors have different approaches to creating portals. Epicentric have elaborated considerably on how portals are created with their system, thus this system is used as illustrations.

Epicentric apparently uses a web based graphical user interface, integrated into the portal, for all operations regarding the creation and administration of new portals.

- *Site Creation Wizard*
  Epicentric employs a wizard user interface to create a new portal site. This is a short procedure, consisting of naming a URL for where the portal should reside, add workbenches, add portlets to these workbenches, applying a theme, and then preview the site. If the site looks good, it can be deployed as a sixth and final step.

- *Shared Components*
  Shared components refer to components that can be shared and exported between portal sites and portal server deployments. They can be used when constructed a new site. Shared components include modules, pages, themes and other user interface elements.

- *Rapid Site Assembly*
  As a new portal site is created, the administrator may pick from reusable, shared components, and can assemble the site using the site assembly graphical user interface.

- *WYSIWYG creation of menus and submenus*
  Epicentric provides a WYSIWYG creation of the menus used by the users to navigate through the portal.

- *Export and Import of Portal Definition, Backup and Staging*
  In addition to visually and with a graphical user interface create and manage the portal, Epicentric provides a feature for the export of an entire site definition. Not only the definition of the portal, but also the portals components can be exported and imported. The definitions are exported as XML files. These files can be imported into the same server as it was exported from, but can also be imported on another portal server. These are very interesting features, and can be used for several functions:

- *Sharing*: Sharing of portals definitions between portal server installations
- *Backup*: Take a snapshot of the portal definition, for later crash recovery
- *Versioning*: If a new setup proves worse than the old one, go back.
- *Staging*: Go through alpha and beta version of a new portal, use test groups. When the portal satisfies the set needs, import the definitions to the production server.

Direct File Handling

BEA have a direct link between one portal and one web application. By copying the entire directory in which the web application "stock portal" resides to a new directory in the same J2EE enterprise application, a new portal is created. This is, judging from the documentation, the proper way to create a new portal.

In this new, empty portal, one can begin to define pages and portlets that should be included. This is done using the stand-alone graphical user interface called E-Business Control Centre (EBCC), which directly edits the XML files used by the J2EE server system. BEA points out that there is nothing hindering the administrator from doing these changes by hand, however, it is considerably easier to use the graphical interface. The portal is first defined and entered into the enterprise application's definition files. Next, elements like skins, layouts, portlets and user profiles are added to the portal's web application definition files. The workbenches are created, and defaults for new users are defined. The portal's definition files are then synchronized with the J2EE server.

After this, further administration is done through the portal server's administration consoles, which are elaborated on in a Chapter 9.3 – "Administration and Delegation".

10.4 Portlet Creation

Not only the portal has to be created and defined, indeed the portlets must also come into existence at some point. Most portals have a set of portlets bundled, at least as examples. In addition, new portlets can usually be downloaded for free or bought at the vendor's websites.

However, for portal frameworks, a good and expressive way of creating new portlets must exist. The different vendors have several approaches to the creation of new portlets. Some will be presented here.

10.4.1 Developing Portlets using APIs

Several vendors have defined some kind of API on which to develop new portlets. By providing a good API, the vendor let the developers use their programming skills to develop exactly what they want, and not restricting the code-logic in any way.

Stand alone vs. Coupled API

The different vendors have selected different levels of detachment of their API. The current Apache Jetspeed API is very tightly connected to the framework, virtually not distinguishing the API from the underlying platform on which it runs. On the other

hand, IBM has made a completely stand-alone API, with the intention of standardizing the API[53].

As a solution lying somewhat in between these two extremes, the Plumtree and Oracle approach is to supply APIs physically detached from the implementation. This is a result of their two-tier approach to rendering of the workbench, confer Chapter 9 – "Architecture and Architectural Features". The two-tier approach severely complicates the interaction between the portal and the portlet, but this is hidden behind the API. However, the two systems using this approach neither have any provisions for one portlet to communicate with another portlet, or an explicit way for the portlet to communicate with the portal server.

Multiple Language APIs

Another effect the two-tier rendering approach is already mentioned, but should be emphasized again: the portlet implementation is separated from the portal implementation by way of a common, standard protocol, HTTP. This implies that the programming language for the portlet is also disconnected, and a portlet can be written in any language. However, the communications between the portal server and the server where the portlet is residing is rather complicated. To hide these intricate exchanges, the vendor supplies multiple portlet development kit (or gadget development kit in Plumtree's case) for several languages. This kit provides an API for the developer to program new portlets. Oracle supplies PDKs for Java, their own proprietary PL/SQL, "URL based" portlets and Web Services based portlets, while Plumtree are more geared towards much used web languages, supplying GDKs for JavaServer Pages, Java Servlets, Microsoft's Active Server Pages and Perl.

10.4.2 Editor Based Portlet Creation

Several vendors have made some kind of visual creation of portlets.

WYSIWYG and Wizards based tools

Oracle have a long time experience with application development using forms and queries, including their large "Oracle Forms" application. This tool is tightly integrated with the Oracle Database, leveraging the PL/SQL language. These tools also provide the ability to web-enable such forms, and place them into the PL/SQL type of portlet provider.

Epicentric provides their Epicentric Foundation Builder. This product lets "business users" create Epicentric modules, Epicentric's equivalent of a Portlet. The tool supplies several Wizard-style step-by-step editors, where a user is lead through the definition and creation of a portlet. Forms can be built with visual tools, letting the user add buttons and modify form properties. Reports are created with a query tool.

Plumtree mentions briefly that they are developing a gadget framework for developing stand-alone portlets, not requiring any underlying data source. These portlets would contain their own application logic and data storage.

[53] For more on the IBM stance, read the analysis of IBM, and Chapter 14 – "Portal Framework Evolvement", which gives more details of how a standard Portlet API will affect the corporate portal business.

Enterprise Application Specific Development Tools

Plumtree supplies a set of Enterprise Class Gadget Frameworks. These are frameworks connecting to a specific underlying enterprise application, Plumtree mentions SAP R/3, PeopleSoft 8 and Siebel Sales. The framework exposes a set of objects representing corresponding objects within the enterprise application. Using these frameworks, a developer is helped through the selection and assembling of portlets using a graphical wizard type step-by-step editor.

Benefits and Drawbacks

Using a visual approach, the vendor gives the developer some benefits. First, rapid portlet creation is achieved. The user can generate a portlet within minutes or hours, instead of days or weeks. In addition, the portlets are very similar, making a uniform user experience. The user can get used to how portlet are working and operating, and all portlets developed within the same framework will look and operate in a similar way.

However, there are drawbacks as well. The developer is limited severely, only being able to use the methods and functionality made available by the framework. Anything outside of the scope of the tool is hard to do, unless the framework have made provisions for the developer to use another development language on places where the framework is too restrictive. So far, the portal systems analyzed did not have such features.

10.5 User / Portlet Communication

The portlet must at some point render its output, and the portal must include this output in the correct place in the output stream. In addition, the portal server must enable the users to direct responses to specific portlets.

10.5.1 Portlet Output to User

At some point in the processing, the portlet must send its output to the user. As all the analyzed portal vendors are using the same approach of rendering the full page for each action done by the user, they all have to use some sequential output inclusion technique. This is quite similar for all frameworks, however, some differences can be seen.

The Jetspeed approach is to invoke a method named getContent() on each portlet. The portlet should then return a string object representing the entire markup that this portlet would like to include[54]. The same approach is in effect used by the two-tier rendering systems, Oracle and Plumtree. There is some overhead involved, as all markup must be generated inside the portlet's code before it is returned to the portal server, thus wasting temporary space. However, this system makes it possible to contain errors. If the portlet crashes while processing the request, the portlet will not have had the possibility to corrupt the output stream yet. The portal server can choose to display an error screen instead of the portlet.

IBM and BEA are instead using a method where the portlet have direct access to the output stream. As the portlet outputs characters to the stream, the client is receiving

[54] The getContent() method is actually supposed to return an ECS element, but the net effect is the same. Review the analysis of Jetspeed to get a fuller picture of this.

these more or less directly, only depending on the buffering present in the underlying systems. IBM is explicitly giving the portlet a handle to the output stream on process invocation, while BEA is using a system of recursive JSP file inclusion, giving the same net effect. This system have the distinct disadvantage that if the portlet crashes while, for example, in the middle of a HTML table rendering, the resulting web page will be garbage. The user will be fed inconsistent markup, which, depending on how strict the browser processes HTML, will result in an incomprehensible web page. This will be especially problematic if the error does not go away by itself, as the portlet is still sitting on the workbench, and will crash on each reload.

Name-Clash Avoidance

With multiple portlets on one web page, possibly several instances of the same portlet, there is a definite problem of *name clashes*. There is a chance that two portlet can both be named the same, and that two portlet developers could choose to name any HTML element or JavaScript variable the same. Therefore, both the id of the portlet and any variables in the outputted markup must be augmented with some instance specific id.

10.5.2 Input from User to Portlet

Addressing

In a normal, full screen web application, the URL that should handle the input from the user is explicitly mentioned. This is true when both using the POST and GET methods of invocation. The invoked URL must also provide the next screen, either directly, or by redirecting the browser to another URL. However, when in a portlet setting, there might be multiple receivers. For example, one portlet might have an input box for quick searching through the internal document archive, while the main portlet on the page is currently showing a form for entering a new customer. In this setting, there are two portlets, of which one will be given the parameters. If the new customer portlet gets the parameters for the search portlet, it will probably choke, understanding neither the parameter's name nor value.

Each vendor seems to have their individual approach to this; however, the basic thinking is the same. Either an extra parameter is used to denote the id of the portlet that should receive the rest of the parameters, or the URL is appended with some information that represents the id of the portlet that should receive the parameters.

Event Driven Programming

By making a system for a portlet developer to make actions that the user fires, the portlet may be developed like an event-driven program. This further strengthens the analogy with an operating system, as most OSes lets their applications react to user events like mouse clicks and keyboard strokes in an the same fashion

The IBM APIs have driven this approach to the fullest, utilizing the same type of event system that the Java APIs are using. Every action or event triggered by the user or the system can be picked up by the portlets by implementing the corresponding listener interface. The methods defined in the interface are invoked when the specific event is fired, providing the object representing the event as a parameter.

Several types of events are defined. WindowEvents are fired when the user clicks on a close, minimize and similar buttons on the window frame to a portlet. The MessageEvent is fired when portlet is sending a message to another portlet, this will be explained further in the next subchapter. By invoking the createURI() method defined in the API, the portlet can generate a special URL that will "fire back" to the portlet when the user activates it. An ActionEvent is fired if the user clicks or otherwise invokes this special link.

10.6 Messages, "Inter Process Communication"

The portal system from IBM seems to be the only one implementing explicit messaging between portlets. Methods are provided in the API for sending objects from one portlet, addressed to another portlet, or optionally broadcast to all portlets on the active workbench. The receiving portlet must on the workbench. If there are several portlets with the same name as the one specified in the method invocation, all of them get the message.

The message is delivered as a MessageEvent on the receiving side, implying that the receiving portlet must implement the message event listener interface as described above.

The concept of messaging is an analogy to the *Inter Process Communication* concept that is found in most operating system; a process may send information to another process using some defined means.

10.7 State, Persistence, Preferences and Configuration

An ordinary web application is usually using the URL line to keep track of which page the user is viewing, thereby providing the current state of the user, much like a state-machine transition table. In addition, parameters kept in the URL can be used to, for example, keep the id of the active customer or the currently highlighted item in a list. Combined, these variables comprise the *state* of the user's experience. If this start to complicate and too many things should be remembered between each invocation of the application, a *Session* system can be utilized. This is a store of named values, which is kept in the server's memory or on disk between invocations from the client's browser. The user is uniquely identified to his specific set of session variables by means of a *Cookie*, or by having the URL rewritten to include the unique ID specifying the user's session. However, the fact that a portal workbench most often contains several portlets again gives rise to challenges compared to full screen applications.

Persistence refers to state variables that are saved between each user session. This means that if the user logs out, and then logs in, the persistent variables in the state will be restored. This is often used to store a user's preferences and configuration concerning each portlet. The user's global preferences are also a part of the user's persistent state, for example, which workbenches the user have defined, what portlets are on each workbench, and the theme selected.

The scope of state variables can be different. A user might have a specific portlet present on several workbenches, and one issue is whether the state should be one for each instance or one for all instances. In addition, the concept of user groups is another place where scope is an issue; a portlet's state could be common for all

members of the group, or each user could have their own state associated with the portlet.

In addition, when administrators install new portlets, they often have to supply some configuration details, for example the hostname of the underlying enterprise application system to which the portlet should connect. This can also be viewed as a type of persistent state variables; however, these are global and persistent.

The different frameworks have solved these issues in different ways, some of which will be highlighted.

Two-tier rendering systems

The two tier systems are in a special situation when it comes to state handling. The problem is that the portlets are residing on separate physical servers, and the question of how to handle each user's state variables, especially the persistent ones, arises. Oracle have chosen to let the portlet server handle this, and the user thus gets one session on the portal server, and one session for each portlet it is using. If the portlet makes use of persistent state, this is stored locally on the portlet server, using either a FilePreferenceStore or a DBPreferenceStore object. The obvious problem here is the one of user management and what happens when a user is deleted, since there is no central place where the user's profile is stored.

Epicentric have solved this elegantly by making the portal server handle all the persistent state. The portlets on the portlet servers can send back attributes that should be stored via the HTTP headers. Next time the portal server invokes the particular portlet for the particular user, these parameters are sent along. The portal server thus acts as a central persistence store, and the user management problem is eliminated.

Single-tier rendering systems

Apache Jetspeed uses a XML file with Portal Structure Markup Language (PSML) to store the preferences for each user. Apache Jetspeed seems to have no clear way of organizing temporary state. However, persistent state is provided by a persistence service, providing one parameter store per portlet per workbench per user. This service lets the developer insert string values on string keys, and then invoke a method to persist the new values. Configuration is done using the Registry, which is a set of XML files.

Again, the clean, stand-alone API from IBM provides the most tidy and centralized way of handling state. The PortletSession provides for the temporary state handling, and is scoped per portlet per workbench per user. The PortletData provide persistent state, and is scoped similarly to the PortletSession, unless the portlet resides on a *group page (workbench)*, in which case it is scoped per portlet per group workbench. PortletSettings provides for the configuration details. These are scoped for all portlets with the same id. A set of settings are always provided with the portlets, however, the administrator may instantiate new portlets, giving them a new set of settings.

10.8 Roles and Permissions

Roles and permissions are explained in Chapter 10.2 – "Security". These properties, which the user either explicitly is given, or implicitly are granted by group membership, can be tested for by way of method invocations on some object. The

only system that directly mentions this way of access control is Apache Jetspeed, which provides the system as a services. This system can be queried of whether a user is in a specific role, and whether he has a specific permission.

10.9 Internationalization

As mentioned, the internationalization capabilities for the systems are not overly impressive. BEA have two special JSP tags involving the internationalization. These are the <i18n:localize> tag, used to define the language, country, variant, and base bundle name to be used throughout a page, and the <i18n:getmessage> tag, used to access the specified resource bundle. The IBM API only specifies a very rudimentary helper method for accessing text from localized resource bundles.

10.10 Caching

As mentioned in Chapter 9 – "Architecture", several of the frameworks have built-in caching support. However, vendors have chosen different approaches to caching. Plumtree is using a declarative approach, where the administrator is configuring the caching behavior of the portlets. Jetspeed is providing a caching service for the portlet developers to use, reducing the implementation burden on the developer. Epicentric provides a caching API, which is letting the developer choose both how the portlet's content should be stored, and by which keys the content should be cached. Finally, IBM is using a HTTP-like programmatic approach, having the portlet implement a getLastModified() method. This method should return the time at which the portlet's content was last modified, thus enabling the portal server to decide if it has to actually run the content generating methods again, or just use the cached information.

11 Software Quality Model

Following is a brief analysis of how the different products as a whole relate to the chosen characteristics of the ISO/IEC 9261-1 Software Quality Model, confer Appendix A – "Definition used from ISO/IEC 9261-1". Several of the elements beneath each characteristic were elaborated on in the preceding chapters, and are therefore just mentioned briefly here.

11.1 Functionality

Interoperability

One of the main arguments for a portal system is the ability to display from, and gather data to underlying systems of diverse types. Choosing only one technology, Java is currently not a bad option, as there are Java APIs for most any computer technology available. Java enables connections to other URLs, to databases, and other resources, communicating via most any protocol.

A portal system should also interoperate with underlying enterprise applications. However, this is not necessarily the portal server's responsibility, but can be done by independent *connectors* that communicate with the application. The enterprise application's functionality is exposed as an API. Several enterprise applications already have some external API defined for some language. In addition, the J2EE Connector Architecture [JSR 16] specifies "… a standard architecture for integrating Java applications with existing enterprise information systems.", thus providing a standard way of enabling Enterprise Application Integration with Java. A considerable amount of vendors has already made connectors for their applications, and third parties are implementing connectors for other systems[55].

Some portal vendors have already developed special connectors towards specific systems. For example, Plumtree have a set of Enterprise Class Gadget Suites, which are pre-made sets of portlets connecting to specific enterprise applications. In addition, Plumtree have made development frameworks for certain selected enterprise applications. These are called Enterprise Class Gadget Frameworks.

The two-tier rendering approach used by Oracle and Plumtree are interesting concerning interoperability. Since portlets can be programmed in any language, portlets can be programmed on the host system's technology instead of using the portal system's technology.

Security

Java as a development platform is not as error prone to common security flaws like *buffer overflows* as many more low-level languages are, for example C and C++. The entire Java virtual machine's architecture is made for restricting a programmer's ability to code dangerously, thus heightening the security level.

[55] J2EE Connector Architecture products: http://java.sun.com/j2ee/connector/products.html (Listing of application servers vendors, EIS vendors and third party vendors)

The portal systems' transport layer, that is, the connection between the user's browser and the web server, is usually protected by the industry standard Secure Sockets Layer, SSL.

Plumtree argues that the two-tier rendering approach heightens security by providing an extra layer between the user and the enterprise application. The portal server is accessing the portlet server, which again is accessing the enterprise application. The system can be set up so as there are no direct route between the portal server and the enterprise application, thereby lessening the impact of a security violation or break in on the portal server.

By providing, or interfacing to, a single sign-on system, a portal server can heighten the security since users do not have to remember and possibly write down multiple passwords. However, this idea can backfire if this single portal password is disclosed, exposing a whole set of underlying enterprise applications.

11.2 Reliability

Fault Tolerance

A portal is a complex system, integrating towards multiple underlying sources that can crash and create problems independently. In addition, each portlet is a part of a full web page, which makes the system sensitive to each portlet's markup language output. Finally, the portal server itself could crash.

BEA and IBM's system is using a system of recursive and sequential portlet content inclusion. If a portlet outputs any bad markup language, either because of a logical bug, or because the portlet crashes midway through its rendering, the resulting page can be incomprehensible for the user's browser. This is especially problematic if the problem does not clear by itself. The portlet will continually crash some way into the rendering, and since the resulting workbench is incomprehensible, the user cannot remove the portlet from the workbench either.

Apache, Plumtree and Oracle use a system where the portlet is supposed to return its entire content in one operation. Apache is doing this using a method invocation on the portlet object, which is required to return the entire portlet content. Plumtree and Oracle are using the two-tier rendering approach, thus insulating the portlets' rendering from the rendering of the rest of the workbench.

It is possible to make a portal server that could embed potentially error prone portlets in a "protection shielding". This could be used on portlets that are in an early stage of development, beta versions, and new, untested portlets. Instead of letting the portlet write directly to the output stream, the portlet output could be temporarily stored before inclusion into the workbench's markup. This is in effect what happens with the Jetspeed's getContent and the two-tier rendering approaches. If the portlet crashes, an error message can be shown instead of the possibly incomplete content from the portlet. It is even achievable to validate the portlet's outputted markup for completeness and correctness before inclusion. None of the vendors does however elaborate on this possibility, but this has been discussed on the Apache Jetspeed mailing lists during the portlet API discussions.

The Plumtree documentation is the only one mentioning this type of error catching. If a portlet crashes on a Plumtree system, the portlet is taken out of service. If the system is set up with multiple servers per portlet, then the portal server can fail-over to another server that is running the same portlet,

The J2EE systems from the commercial vendors are also often fault tolerant, in that it can have fail-over capabilities. This means that the system is set up in a clustered fashion, where several J2EE servers are working in unison. If one of the servers fails, this is picked up by the remaining servers. Depending on the configuration, this can be made to happen totally transparent to the connected users.

Recoverability and Availability

Since the portal will be used by the entire corporation, centralizing much of the enterprise application usage through the portal, it is imperative that the system is available at all times, and that it can recover fast and well in case of failure. Still, only Plumtree specifically mentions recovery and availability in their portal documentation. Plumtree lets administrators run their portlets on multiple servers. If one of the portlet servers goes down, the portal server automatically fails over to another available portlet server that is running the same portlet. When the portlet server comes back on line, the portal server automatically starts to use it again.

Nevertheless, several of the commercial application servers provide "high availability" system. As an elucidating example, the Oracle 9i Application Server has multiple features with the specific goal of keeping the system up, running and available. The system is designed with a *no single point of failure* strategy, and connections are *automatically rerouted* around failed elements of the setup. *Transparent application failover* keeps the users of the system unaware of failing server instances, while the *automatic death detection* combined with the *fast restart architecture* takes care of getting failing instances of the system quickly up and running again if a failure should occur. [Oracle A]

11.3 Maintainability

Analysability

The BEA system contains a feature where each JSP and servlet invocation counted and timed. The administrator can easily check if some part of the portal suddenly is using much more time than usual, indicating some failing connection or similar problems.

The Apache system has no clear distinction of what is the framework and what is the API, making analysis and bug hunting a more problematic process. On the other hand, IBM's Portlet API is very clean, and makes a clear distinction between the portal server and the portlets. This helps a developer to isolate where errors occur, as the invocations of the portlets methods are clearly defined.

The BEA system of recursive inclusion of multiple JSPs for each portlet, and the workbench as a whole, makes for tricky debugging. If a erroneous element is discovered in the final workbench rendering, it can prove difficult to find which JSP file that actually is responsible for that exact part of the markup language.

Changeability
Since the analyzed systems are frameworks, their primary concerns are, of course, changeability. The main idea is to let administrators and developers install, configure and develop portlets. Enterprise Application Integration, data source communication and user management is also crucial. In addition, most systems also let the look and feel of the system be changed by means of themes or skins.

The Apache Jetspeed system is open source, and is therefore changeable in every way, even the framework itself. BEA's system is also quite open, with most of the portal developed on the same vendor's J2EE server. The JSP files are available, as are the webflow comprising the portal's operations, which means that it can be changed in most every way.

In contrast to these, the IBM documentation currently available only concerns portlet development. No information about themes, system look and feel and the rest of the system management is available.

Stability
A portal system should *ideally* handle that a *malicious* portal application was installed, whose portlets tried in every way to make the system crash. This is not the case in any of the tested system, instead, any portlet on all these systems can way to easily crash the entire system. In addition, they have access to the entire system.

The Apache Jetspeed system is in beta revisions as of this date, and is not considered ready for production usage. The BEA system is developed on BEA's existing, well-tested technology, and should therefore be rather stable. The rest of the systems are rather new technology, and no information about their stability is available.

Testability
The modular nature of a portal system should enable modular testing, where each unit is tested by itself, verifying its function. This is especially true for the IBM system, where the stand-alone portlet API could enable the development of a testing harness in which to run and test the portlets. However, overall the testability support was quite disappointing. No system was found to have test bench systems for single portlets. Oracle mentions some features in their developer environments, but these have not been delved into.

The BEA system has access and timing statistics, as mentioned under the "Analysability" section. This can help as a first step in debugging a portlet.

11.4 Portability

Adaptability
There are two areas in which these systems show adaptability: internationalization and multiple device types.

The internationalization support is, as mentioned, disappointing. While most systems state that they are "fully internationalized", the documentation revolves around languages. Plumtree and Oracle's systems choose language based on the user's

browser setting, while the Epicentric system have a locale setting in the user profile that the user may select. Only Plumtree and Epicentric mentions anything besides languages, and the elements mentioned are dates, times and international character sets, leaving out numbers, currencies and text direction.

Since the portal's job is to make available a host of different internal data sources, support for multiple markups would enhance its value as an information center several-fold. This has also been taken into consideration with at least the systems from Apache, IBM and Plumtree. All of these systems let the portlet take care of the formatting of the portlet's output, while the portal takes care of making the workbench into some format that is appropriate to the markup language and device connected. The reason for not trying to make some automatic conversion between markups, or from a common markup, is stated as the diversity of devices and the markup languages they support. The most commonly supported markup is WML for WAP devices, while Plumtree also supports Palm PDAs and RIM Blackberry pagers.

Conformance

As the portal server is a middle-layer element, it has several directions in which there might be standards to conform to: server on which it runs, the API which it provides, and the way in which the system integrates towards enterprise applications.

All systems except Plumtree are written on the Java 2 Enterprise Edition platform. Plumtree is written in some natively compiled language, and is available only for the Microsoft and Solaris platforms. However, the portal systems delivered from vendors with existing J2EE systems, all require the vendor's own J2EE system. This can easily seem like some kind of vendor lock-in tactic.

The Apache Jetspeed system does not require the full J2EE environment, as it can run on a simple servlets container. Epicentric is a dedicated portal vendor, and does not have their own J2EE server. This means that Epicentric must be run on another vendor's J2EE server, and Epicentric conversely emphasizes the amount of different configurations on which their system can run.

The portlets should ideally be able to run on multiple vendors. However, there are still no standard for portlets. There are several standardization efforts being worked on as of this moment, these are described more thoroughly in Chapter 13 – "Portal Framework Evolvement".

Replaceability

This is a characteristic not quite applicable to portal systems as these systems are supposed to make enterprise applications accessible through an easier interface, and not to actually replace the existing application.

However, productivity applications like calendar systems, communication applications like e-mail clients and front office applications like sales automation system can potentially be implemented directly as a portal application. These applications would then replace other such systems.

As for vendor lock-in, the analyses show that vendors use the portal as another way to leverage their own solutions, instead of trying to make portal systems interoperable.

One exception from this can seem to be IBM, where IBM developers have tried to make a common portlet API with the open source group Apache.

11.5 Scalability

All of the J2EE systems provided by the analyzed vendors are scalable, enabling servers that can run on multiple physical servers. The Epicentric and Apache Jetspeed systems needs a application server to run on, however, since they are standards compliant, they inherit the scalability from the server on which they run.

The Plumtree system lets the developer choose whether each portlet should be run on a shared server, alone on a physical server, or spread out between multiple physical servers. This feature makes the administrator able to put additional physical resources into the portlets that are resource demanding. The system can thus be scaled in a module-oriented fashion. However, the portal server itself does not seem to be scalable, and can thus become a system bottleneck.

12 Portal Framework Evolution

This chapter will try to give some predictions about what is likely to occur in the corporate portal framework area in the next few years.

12.1 Standardization Efforts

The one definite prediction is the emergence of portal and portlet standards. There are two areas where such standards are emerging, namely for the Java programming language, and for the Web Services set of protocols. The first are exemplified by, for example, the IBM Java based Portlet API. The other set are exemplified by the Oracle way of two-tier rendering, using SOAP/XML as the transport language.

12.1.1 Consequences for not having a standard

As there is neither any monopoly nor any standard on the portal market, companies will be reluctant to develop serious applications for portals, thus hampering any progress in the portlet market. A vendor might decide to develop an application or an enterprise integration application for one vendor's platform, only to find that the chosen technology loose momentum just months after the application is finished. This would render the invested time and resources worthless.

The same goes for the enterprise application vendors. As long as there are no standard, these vendors will more likely develop "portalifications" of their own products, instead of developing application integration portlets that could be embedded within any standard portal.

This will continue until either a monopolist is emerging (as has happened with the desktop operating system market, where the Microsoft OS is a de-facto standard), or a common, adhered standard emerge.

12.1.2 Java based Portlet API

Initiated shortly after this thesis was begun, the emergence of a Java portlet API standardization was unavoidable. Java seems to have a definitive market share on the application server market, and many of the portal vendors are already dedicated to the Java 2 Enterprise Edition platform. The standardization process is called Java Specification Request 168 [JSR 168]. This standard's description is, "The Portlet specification will define a Portlet API that provides means for aggregating several content sources and applications front ends. It will also address how the security and personalization is handled". The need of the Java community that will be addressed is, "This specification will establish a standard API for creating Portlets, thus avoiding locking in Portal developers in a specific implementation and allowing Portlets developers to reach a wider audience while reducing their development efforts".

The Java API standardization process is described in some detail in the analysis of the IBM portal, Chapter 8.4 – "IBM – WebSphere Portal". It suffices to iterate that the standardization is in progress, and that the schedule states that by December 2002 the portlet API should be finished, and a reference implementation of the standard should be finished.

12.1.3 Web Services Portlets

The Web Services concept has gotten considerable attention lately. Several Web Services Portlet standard initiatives are currently underway, all recently started. Promising interoperability between different architectures and platforms, a Web Services based portlet standard will give considerable leverage for the portal concept. The following subsections will briefly explain the concept of web services and how web services portlets could work

Web Services

Web Services are defined by the World Wide Web Consortium, W3C as, "A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols[56]".

The Web Services are a remote procedure call (RPC) protocol based on XML over HTTP. It is using several protocols for discovery, localization and description, however, the major protocol facilitating the actual communication is the Simple Object Access Protocol, SOAP.

Web services is a new technology. XML itself was specified February 10, 1998[57], while the protocols that make up the web services framework still are settling. The W3C have initiated a Web Services Activity, where "The goal of the Web Services Activity is to develop a set of technologies in order to bring Web services to their full potential"[58].

To illustrate the Web Services concept, a standard example of a currency exchange application utilizing a web service will be described:

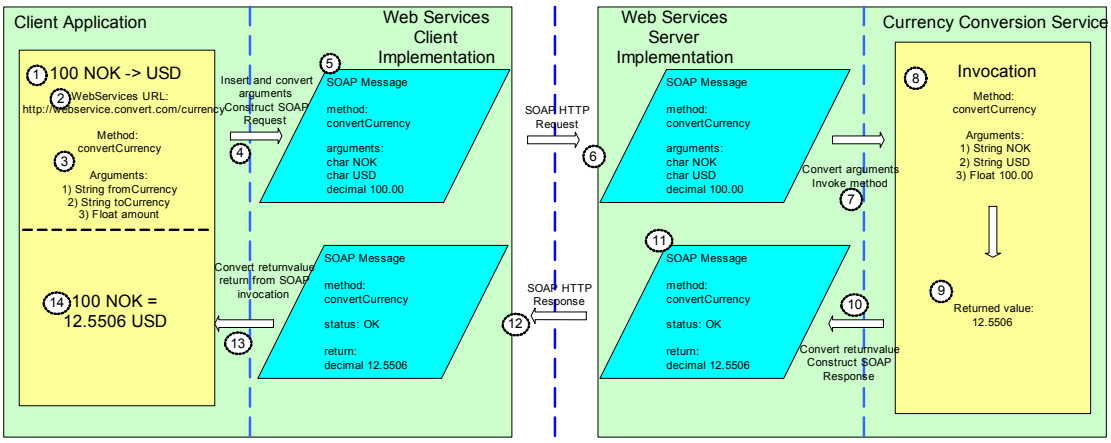


Figure 38 Example of currency conversion using Web Service. Note: this is an abstraction of how a client communicates with a web service, and is not meant to represent the actual messages passed.

1. The application needs to make a currency conversion, and is programmed to use a web service

[56] Web Services Architecture Requirements: http://www.w3.org/TR/wsa-reqs
[57] XML 1.0 W3C Recommendation: http://www.w3.org/TR/REC-xml
[58] Web Services Activity: http://www.w3.org/2002/ws/

2. The client locates the URL to the web service using some means, assume hard coded location.
3. The client knows the method names and arguments that the web service takes, called the web service's interface, using some means. Assume hard coded interface description.
4. The client translates the parameters needed for the conversion request to the format required by SOAP.
5. The client constructs a SOAP request specifying the method name on the remote server, in addition to all the necessary parameters.
6. A HTTP Connection is established with the remote machine, using the Web service URL, and the SOAP request is transferred.
7. The web services implementation on the server side translates the arguments in the request to the native language used in the web service.
8. The server invokes the method specified in the request, supplying the translated arguments.
9. The method returns the return values.
10. The return values are converted back to the SOAP specific types.
11. The server constructs a web service response
12. The response is communicated back to the client using the still open HTTP connection.
13. The client decodes the response, and translates the return value back to the client's native language.
14. The application now have the correct value.

"Visual" Web Services

Web services enables two applications, residing on different machines, possibly on different architectures and programmed in different languages, to communicate. The Web Services concept does not have any functionality for a person to communicate with a Web service. For example, the application described in the last subsection still needs to be implemented as a full application on the client side, even if its only job is to take a value from the user, invoke a web service, and display the returned value to the user.

If there was some markup language that could describe which elements a user interface for the web service should contain, what operations that could be performed and which states it could be in, a general visual web service client could be developed that displayed these elements, and let the user choose the operation to invoke. This is exactly what multiple standardization efforts are doing right now.

Compared with the two-tier rendering approach used by Plumtree and Oracle, this approach is very similar. The Oracle solution indeed uses the Web services protocol SOAP to communicate between the portal server and the portlet server.

Standardization Efforts

The XML standardization body Oasis has two standardization efforts running regarding visual web services. The first, most general, are the Web Services Interactive Applications, [WSIA], formerly known as Web Services Component Model [WSCM]. This committee aims at developing a standard for "…interactive, human-facing, Web services". The second, more portal specific, are the Web Services for Remote Portals [WSRP], initiated in March 2002. This committee's purpose is

"Defining an XML and Web services standard that will allow the plug-n-play of visual, user-facing Web services with portals or other intermediary Web applications".

Interestingly, IBM are apparently deeply involved with these standardization efforts as well as the standardization effort on the Java Portlet API, confer the IBM analysis. IBM are currently developing two different standards, one called Web Services for Remote Portals [IBM-WSRP], which is submitted to Oasis, and another, called Web Services Experience Language [IBM-WSXL]. This latter one is apparently designed to be a superset of the WSRP standard. Both the Oasis WSIA and WSRP are chaired by IBM persons, WSRP is indeed chaired by Thomas Schaeck, the same person that promoted the creation of a standard Java Portlet API on the Jetspeed mailing lists, as described in the IBM analysis, Chapter 7.4.1 – "Historical Review of the Portlet API".

In addition, as mentioned in the analysis in Chapter 8.3 – "Epicentric – Epicentric Foundation Server", Epicentric have started a standardization effort called Web Services User Interface, [WSUI]. Version 1.0 Working Draft was released February 11, 2002 and is accessible at the web site. This has now been presented to the Oasis WSIA and WSRP committees as presentation documents.

Technical Description

The idea behind the Web Services for Remote Portals are that the portal server merely instantiates portlet proxies for the remote portlets, and configures these to contact the appropriate URL [IBM-WSRP-WSRP]. When a user puts a web service portlet on his desktop, an instance of the portlet is created on the remote side, along with an optional session. In return, the portlet proxy get a handle to the remote instance with which it refers to this remote portlet instance. When the user removes the portlet from his workbench, the instance and session is destroyed on the remote portlet server.

State and preferences may be stored on the portlet server, as with the Oracle portal system. However, the portlet may also ask the portal server to keep state, as with the Plumtree portal system. The state is then serialized and sent back to the invoking portal server for storage. When the portlet is invoked again, the state is sent to the portlet server again, confer the Plumtree analysis. The portlet can also use a combination of these, where some state is kept locally at the portlet server, and some state is sent to the invoking portal server.

When the user clicks some portlet link, or submits some data, the portal server translates this to an action that is invoked on the remote portlet. The parameters are submitted, along with the action id, on which the remote portlet acts.

It is important to note that portals based on such web services portlet are not bound to be implemented in any specific language. This can be seen by the already existing portals using a similar approach, one most likely written in a natively compiled language (Plumtree), another is using a hybrid between native compilation and Java (Oracle with their application server and J2EE environment), while the third is written entirely on the J2EE platform (Epicentric, J2EE platform, using Modular Web Services).

WSRP and JSR 168 Synergies

A potential exist for synergies between the web services portlet and the Java Portlet API standard. These standards can potentially be made in such a fashion that a portlet developed in the Java API can transparently be made into a web services remote portlet by simply configuring it as remote. The IBM presentation to WSRP submissions points out such a possibility [IBM-WSRP-WSRP], as does the JSR 168 presentation[59]. The JSR 168's charter mentions, "It is an important goal that the design of the Portlet specification would allow implementations to support remote Portlet execution".

As mentioned, the chair of the WSRP Technical Committee is Thomas Schaeck from IBM, which also acts as IBM's contact person on the JSR 168. Likewise, Sun's specification lead of the JSR 168, Alejandro Abdelnur, is listed as Sun's member of the WSRP Technical Committee. It thus indeed seem like the communication paths between the WSRP and JSR 168 are established.

12.1.4 Possible features included in the standardizations

Based on the preceding analyses, some probable high-level features and concepts can be listed for the portlet API and Web Services Portlets standardizations. These are concepts and features that seems very interesting to include in a common standardization effort, but which have not been implemented by all vendors.

Themes

Themes control is a definite must, and is included in all systems. However, the fonts and font sizes and colors seem to be neglected in most systems. For HTML, a standard set of Cascading Style Sheet styles-definitions and color variables should be defined, so that themes easily can be implemented. A theme would, in addition to workbench layouts and similar high-level design features, also define the standard styles and colors. The entire portal design can thus be changed, enabling the use of high contrast, inverted colors and large font themes. This has been taken care of in the WSUI standardization effort [WSUI].

Stand alone portlet applications

At least IBM has implemented this architectural feature, but the providers concept from Oracle also bundles several portlets in one package. Portlet applications will enable portal administrators to install new applications into their portal, much as an application can be installed on an ordinary operating system.

Services

For the Java based Portlet API, a standard way of extending the API by means of services seems like a logical addition. The IBM API provides such functionality.

Dependency control

Portlets, portlet applications and services should be enabled to register that they provides certain features. Other portlet applications can then define that they require, or depend on such features, this to assure that the installation of, for example, a given

[59] JSR 168 presentation for WSRP TC: http://www.oasis-open.org/committees/wsrp/presentations/jsr168-wsrp_presentation.ppt [Accessed June 4, 2002]

portlet application will succeed. Such features are needed once multiple applications that possibly will communicate with each other can be installed and removed by merely clicking the mouse button. Dependency-control systems will warn an administrator that is trying to install an application that needs the features of another application that is not present, or trying to remove a package that is needed by other, still installed applications. Such features are provided in most UNIX operating system, for example the Redhat Package Management system, rpm[60].

Roles and permissions

A portlet, being it a web service portlet or an API based portlet, should be able to define multiple permissions associated with its functionality. The administrator will use the roles editor to include the permissions defined by the application into the different roles defined in the portal, or possibly create a new set of roles to handle these new permissions. This is adopted in the Apache Jetspeed system. The security level system of Oracle is a crude implementation of the same idea.

Standard portlet sizes

One fact that seems to be neglected by all systems is the reality that the strict layout defined in most portlet system, to a large degree confines the amount of information that can fit into a single portlet. Specifically is this important regarding the width of the portlet, as the typical small portlets for example cannot contain several input fields horizontally. A set of portlet sizes should be defined, so that a portlet can be assured that its content will fit into the column in which it is placed. The portal would not let a user place a “medium” portlet into a “small” column, as this would destroy the layout. Another way of doing this, especially regarding portable, small devices, is to enable the portlet to know how large the screen size the attached device have, for example in characters or pixels.

Internationalization

The apparent lack of focus on internationalization must be addressed. Portals are useful for most people, regardless of which culture they come from. A user should be able to specify in which language he prefers his user interface, and how dates, times, numbers and currencies should be formatted. In addition, his default currency and time zone should be selectable. Such a setup would for example enable a Swedish salesman that is working for a Norwegian company and attending a conference in Californian, to adjust his date, time, number and language preferences to Swedish, his default currency to Norwegian, while adjusting his schedules and other activities to the Californian time zone. The APIs defined should include a vast amount of helper functions for such formatting, as to make the coding of internationalized portlets convenient.

Security contained portlet applications

The Java environment enables a java class file to be run in a so-called security sandbox. This could be implemented in the portlet APIs and pertaining definition files so that beta versions or distrusted portlet applications could be installed and tested, without the risk of compromising the corporation’s data. A buggy portlet could potentially erase, destroy and corrupt data and files from the underlying operating system and system database. A malicious Trojan portlet could, if left unchecked,

[60] RPM website: http://www.rpm.org/

wreak havoc on the enterprise network. It could potentially download and execute code, read or destroy files and other resources, make unhindered connections to any server on the enterprise local network and even make use of locally exploitable security weaknesses of the hosting operating system, all while giving reports back to an outside host.

12.1.5 Consequences

A standardization of a portlet API and web services portlets would lead to many new possibilities and opportunities. All companies that install a standardized portal system would instantly be potential users of both locally installed and remotely served portlets. Some likely effects will be discussed briefly.

Stand-alone Portal Applications

The portlet API standardization would make it interesting for several parties to make small and large portlet applications. Plumtree states in their February 2002 customer survey [Plumtree A] that "Nearly half of survey respondents have developed ten or more graphical web services for their portal deployment, and 12% have developed 50 or more". This included both enterprise integration applications and stand-alone portlets, however, most were in the latter category.

Several software companies will use the opportunity to make stand-alone portlet applications that does not depend on any underlying application. Such applications include all types of database-near front-office and productivity tools, for example group calendars, project management systems, collaboration tools, and sales automation and customer relationship tools

Enterprise Application Integration

Software vendors will see an opportunity of making enterprise application integration portlet sets, where a portal administrator simply buys and installs a portlet application tailored to the company's chosen enterprise application. This will enable all of the company's staff to access the data contained in the enterprise information system. This can already be seen with their enterprise class gadget suites and frameworks. However, if a portlet standard is established, this will be a viable way for several companies.

In addition, the enterprise application vendors will themselves make commonly used subsets of their system accessible from a set of portlets to increase the usability of their product. This is highly interesting when a standard way of creating such applications is made available. Web services portlet standardization will make this even easier, as the enterprise application can be embedded with a web services server, and publish its integration portlets by default. The portlet administrator only needs to point his portal to the URL at which the services are published, and instantly the enterprise application is made available to all employees.

Move Customer Portal over to Customer's Portal

Another shift made possible by the web services portlets is that for example customer self-service applications may be moved closer to the customer. Rather than giving a customer a username and a password to the seller's customer portal, the customer can instead point their portal to the seller's web services portlet, and thus integrate the seller's order forms and similar application directly into their own portal. Other

possibilities are all type of applications that usually are hosted on a separate web site. All examples of external resources given in Chapter 6.2.1 – "Make Information Accessible" can be accessed via the portal as they were local resources, significantly impacting the way that users do work. This concept has been touted by Epicentric with their Modular Web Services[61].

However, one major hindrance to the full utilization of this potential is the web service's current lack of security and authentication standards. There is currently no common way that the seller can authenticate the customer, or for the customer to authenticate the seller. This still makes it necessary for the seller to issue some kind of credentials, and for the portal administrator to configure the portlets with these credentials. Oasis has initiated an effort to standardize web services security, the Security Assertion Markup Language [SAML]. This specification is soon to be released. Java Specification 155 [JSR 155] aims to implement this specification. Another standard regarding this issue was initiated recently by IBM, Microsoft and Verisign [IBM-WS-S, MS-WS-S].

Portlet Development Kits, WYSIWYG Portlet Creation

As the standards emerge, it will become viable for third parties to develop portlet development kits and "studios". All the large standard technologies, like HTML, ASP and JSP, a vast set of development tools exists, for example FrontPage from Microsoft and Macromedia's web development tools. Such tools will definitely be made for the portlet technology if standards are agreed upon.

These tools will be made for both the portlet API and the web services portlets. For the portlet API, tools will emerge that helps developers in creating portlet and portlet applications in WYSIWYG environments, as can be seen in the full screen HTML tools mentioned. However, an even larger potential are the web services portlets standardization, as this will enable portlets to be written in any language, and hosted on any portlet platform.

Some portal vendors already provide wizard-driven and visual style portlet editors, confer Chapter 10.4.2 – "Editor Based Portlet Creation", however, there are currently no third party tools facilitating such portlet development.

12.1.6 Potential Difficulties

The standardization holds great promise for portal systems. However, not all analyses of the standardization conclude that the full potential will be achieved soon. An article by David L. Margulius [2002] points out that the corporate portal area is a "battle over the heart of the enterprise IT stack" as portals begins to be the focal point of any organization's IT strategy. The push for a portlet standardization is in effect coming from the customers, driven by the high costs of portal implementations. However, the vendors will not really be interested in an actual standardization. Standards would commoditize their offerings, and thus enable other companies to exploit their application logic and infrastructure functions, giving them less leverage for their application servers and infrastructure products.

[61] Epicentric gave an example of this usage; however, the web page in question is now removed from Epicentric's site. Epicentric has lately moved away from the "Modular Web Services" name, and the concept is now called "Epicentric Web Services". The example used was a financial services institution that developed a modular web service. This financial institution's customer was using the Epicentric portal, and could therefore import the financial institutions modular web service into their portal by instantiating and configuring a MWS Building Block.

Margulius predicts that a likely outcome of these standardization efforts is that basic interfaces between portals and portlets are defined, however, the higher-level interfaces for cooperation such as sessions or process flow management will still be proprietary. As Margulius states it, “In other words, we’d get neater view into the same old bunch of silos.”

Another problem that the portal industry faces is that the number one desktop operating system vendor Microsoft has not joined any of the portlet standardization efforts. The reasons for this can seem obvious; Sun and Microsoft have had several legal battles over the Java platform, resulting in Microsoft’s complete abandonment of the environment. In addition, IBM is in important positions in both standardization efforts, thus linking the web services and Java standards more than desired by Microsoft. IBM is in addition a long time Java supporter, making the problem even bigger.

This fact can hamper the standardization efforts to a considerable degree. Microsoft’s complete control over the desktop environment means that whatever they decide upon will be very influential for the entire IT business. With the release of the .NET platform, the company can potentially lead the customers and developers away from the platform-agnostic concept of the corporate portal. More on .NET is found in Chapter 12.3.1 – “Microsoft .NET”.

12.1.7 Vendor Diversification

Given that standards appear, a natural question is to ask how vendor will diversify and get business. This question is especially interesting since there exists a free reference implementation of the portal system.

The J2EE 1.3 specification mentions this in Chapter J2EE.2.9 [J2EE], “We expect J2EE products to vary widely and compete vigorously on various aspects of quality of service. Products will provide different levels of performance, scalability, robustness, availably, and security.” However, several other areas are of interest.

User Management

The user system is left out of any standard. Whether the user authentication and other features of the user is obtained from an LDAP source, or from other sources, are not mentioned. The installation and administration of the authentication system may have different levels of user friendliness and features.

Groups, roles and permissions will most probably be included in a standardization effort. This must also be taken care of by either a disjoint system, or included in the authentication system.

Portal Specific Variations

It is important to remember that the standardization exclusively will deal with portlets, and not the portal implementation. The portlet API and the web services portlets will only standardize the actual portlet interfaces. The technology running and containing the portlets are not influenced by this, other than being required to support the interfaces defined in the standard. Outside of the standard, several areas exist where vendors can focus their attention.

Firstly, the architecture of the system will most probably be vastly different from vendor to vendor. Some will include the portal as an integral part of the application server they already supply, while others will produce it as an add-on to such application servers.

The entire user interface will not be defined in any portlet standard. A portlet's only requirement is that it is contained within a web page. How that web page looks outside of this, is entirely up to the portal implementer. The way in which workbenches are laid out, and how look and feel and theme support is implemented is also outside the scope of any portlet standardization. This also implies that the workbench configuration interfaces will vary.

Finally, service in general, is an important point. This not only relates to support personnel and agreements available, but also to what additional applications, toolkits, portlet development features and documentation the different vendors supply.

12.2 Enterprise Operating System

During the analyses, an impression of the portal platform as an enterprise wide operating system has developed. This might come more and more into focus as the portal standards mature. In addition, this system will be enabled to collate "applications" as portlets from multiple operating systems, using the developing standard WSRP. The benefits of using such a type of operating system are bountiful; some are reviewed in the following sections.

Client System Independence

As an ordinary web browser is used as the main interface between the user and the application, operating system independence is achieved. The browser is based on a defined standard, HTML. This standard is not owned by any corporation, but is instead developed jointly by a multitude of vendors, single persons and organizations. Even other types of devices, as PDAs and other handheld devices, and television sets, are getting a browser implementation embedded. Several full-fledged open source browser projects exist, most notably the Mozilla browser[62]. This ensures that a standards compliant browser can be implemented for a specific hardware platform in a short time.

Client Device Independence

Portal system standardization efforts are including multiple device support in their activities. This will ensure that a corporate portal can be reached from anywhere, anytime.

Server Architecture Independence

The Java based portlet standard is inherently architecture independent, as Java can run on most any platform. However, the web services based portlet standard makes for even more independence, as both the portal server and portlets can be programmed in any language on any platform. In this scenario, the standard regards the interface between the portal server and the portlets, thus enabling any standard web services portlet to be plugged into any standards compliant portal server.

[62] The Mozilla Project: http://www.mozilla.org/

Spans Entire Corporation

All the employees of the corporation can be given user accounts on the corporate portal, and can thus access the system from any location, using any operating system on any device. This means that applications that are installed on the corporate portal are accessible to all users at the same time. Not only are installation costs reduced, but reach is also increased. Ordinary applications have to be developed or at least compiled for each operating system is should run on, while a single installation is enough to reach the entire corporation when using a portlet application.

Easy Implementation, Reduced Time to Market

The standardization of the API and the technologies surrounding the WSRP standard will make the development of a portlet or a portlet application very easy. A need in an organization can be satisfied very fast, and the development can occur in any language if using the WSRP approach. The time from an idea is conceived, to the time a beta version is available to potentially all the user in the enterprise will be reduced considerably.

In addition, the WSRP approach enables an unprecedented approach to application service providing. If an organization wants to test or rent an application hosted with an ASP, the only operation needed is to configure the portal system to include the WSRP URL. Immediately, the application is available to the organization's users.

Both these methods would obviously require the portal administrator to restrict how should get access to the newly installed application, and if applicable, which access the individual people should get. This would be done using the portal's access control lists, giving access to, for example, the standard beta-testing group, and using the permission control system, including the different permissions into already defined roles, or making new roles.

12.3 Other Technologies

The focus in this thesis have become Java and Web Services. However, there are other technologies out there too. The new platform from Microsoft, .NET, is a clear competitor to the Java platform. In addition, several other more or less important technologies exists.

12.3.1 Microsoft .NET

A full discussion of Microsoft .NET is outside of the scope of this thesis, however, an outline is elucidating. The Microsoft .NET platform[63] is currently being touted heavily by Microsoft as the technology for the future. The core platform resembles Java to a considerable degree. Microsoft has developed a new language for the .NET platform, called C#. This language is amazingly similar to Java. In addition, all code is compiled down to an intermediate form, called Microsoft Intermediate Language (MSIL), this step also resembling the byte-code of Java. To run a .NET program, the Common Language Runtime (CLR) is required. This is a runtime environment much like the virtual machine of Java. However, Microsoft have made the CLR flexible, so that multiple computer languages currently can run on the same virtual machine,

[63] A explanation of the .NET platform is found in this Microsoft article: "Microsoft .NET Framework Delivers the Platform for an Integrated, Service-Oriented Web", http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/itsolutions/net/evaluate/netframe.asp

currently about a dozen languages have been ported with several others on the way. The languages can be intermixed, letting a program developed in a language use objects from another language. Even cross-language inheritance is supported, so that an object for example could be written in C#, subclassed in C++ and instantiated in Eiffel.

This environment and technology is being subjected to standardization through the ECMA standards body[64], and is currently available for both the Microsoft and FreeBSD Unix platform, the latter being a proof-of-concept port for the ECMA standardization[65]. An open source implementation of the .NET development platform, called Mono, is underway[66]. This effort is trying to bring the .NET platform to the various Unix platforms.

The .NET framework is heavily geared towards Web Services. Microsoft is a member of the Web Services Interoperability Organization[67], and co-developed several of the standards laying the ground for Web Services jointly with IBM and other industry leading vendors.

The emergence of a .NET-based portal seems unavoidable. The development of a portlet API in the C# language also seems likely. However, the emphasis on web services in the .NET framework makes it quite possible that the web services based portlets, as embodied in the WSRP standardization, are the way that Microsoft will tout as the correct one, if any. This is based on the strong belief in programming language freedom that Microsoft shows, and which also is a consequence of the web services approach, as explained earlier.

12.3.2 Other Languages

Other languages frequently used in web development include Perl and PHP. An interesting open source portal project built using PHP is PHP-Nuke[68]. This system is mainly an "advanced content management system", but it contains many add-on modules, for example discussion forum, mail client, news federation, calendar and so on. The project has had a so-called code fork, and another project called postnuke has been started[69], providing even more modules. The web site Slashdot[70], popular amongst computer people, are built using PHP, and indeed have the concept of *slashboxes*, portlets-like windows that can be put on the single workbench a user have. The website's software, Slash, is open source[71]. There are however no significant amount of portal efforts in other programming languages.

12.3.3 Non-browser Based Portals

Java was once touted as the write-once, run-anywhere language for every operating system. However, this has proved very difficult for graphics user interfaces, client side. Given more time, such a language and environment will probably come into existence, as several platform-spanning languages already exist. This will clear the way for non-browser based portals. Such a portal can be envisioned as some sort of

[64] C# and Common Language Infrastructure ECMA standardization : http://msdn.microsoft.com/net/ecma/
[65] The Shared Source CLI, which builds on both Windows and FreeBSD, is available at: http://msdn.microsoft.com/net/sscli/
[66] The Mono Project's website: http://go-mono.com/
[67] Web Services Interoperability Organization's website: http://www.ws-i.org/
[68] PHP-Nuke's website: http://phpnuke.org/
[69] postnuke's website: http://postnuke.com/
[70] Slashdot "News for nerds, stuff that matters": http://slashdot.org/
[71] Slash: http://slashcode.com/

standardized virtual operating system, in which smaller or larger stand-alone applications are installed, exactly in the same way as have been described with java based portals using a web browser as user interface. The virtual operating system could handle user authentication, roles and permission, domain wide common data storage and security.

The user interface of such a system could look and be operated much more like a proper operating system, with variable sized windows that could overlap. The controls could be more elaborate, with much more interactivity on the client side. As an example, a seller could let the customers install an application in which they graphically could check out the merchandise, and for example doing calculations of dimensions. When the customers have found the item they were looking for, an order can be placed directly in the same application. The application would contact the seller's server, accessing a web service in which the order was placed, giving instant order confirmation to the customer. The company objectBOX have developed such a product, the INSISO webtop[72].

[72] objectBOX: http://www.objectbox.com/, INSISO Webtop: http://www.objectbox.com/eng/Products/webtop.shtml, INSISO demo installation: http://www.insiso.com/

13 Conclusions

The objective of this thesis was to investigate the functionality of corporate portal frameworks, in particular what such systems provide in terms of functional components, architecture and architectural features, and what concepts and features they include. The thesis furthermore aimed at predicting how portal frameworks are likely to evolve over the next few years.

The findings of the study suggest that different frameworks often have similar concepts and features, due to the portal concepts that heavily guide the overall function of the portal. However, the study also observed significant differences. Architectures are rarely identical, and in particular, no system allows a portlet developed for one vendor's framework to be run in another vendor's framework.

Possibly the most promising observation in regard to future trends concerns the two main standardization efforts that have been initiated. These efforts hold the potential of providing corporations with a novel type of architecture independent, enterprise wide operating system, in which portlets can be developed for a single standardized portal platform, and thus be run on any vendor's portal system. The installation of a portlet application on a corporation's portal will provide all of the corporation's employees with instant access to the application's features. The development of complete, independent portlet applications is likely to become a new industry.

This standardization process will not take place without complications, however. Many industry-leading actors have already joined the efforts, suggesting that a standard would gain wide-spread adoption. However, Microsoft, the leading desktop operating system vendor, has chosen not to participate in any of the efforts, thus potentially constituting a major obstacle to the corporate portal's potential of becoming the new corporate operating system. Additionally, doubts are raised regarding whether application server vendors really are interested in a complete standardization, as this potentially would lessen their leverage in pushing their entire technology stacks onto the customers.

14 Appendix A: Definitions used from ISO/IEC 9261-1

The two standards bodies *the International Organisation for Standardisation*, *ISO*, and *the International Electrotechnical Commission*, *IEC*, has in the field of information technology prepared a *Joint Technical Committee* called *ISO/IEC JTC1 Information Technology*. This committee prepared the standard ISO*/IEC 9261, Information Technology – Software quality characteristics and metrics*. Part 1 of this standard is *Quality characteristics and sub-characteristics*, and some definitions from this part are used in this thesis.

The standard's focus is towards assessing a software product's usability in relation to a specific use or usage scenario. This is especially obvious in the fact that all the characteristics and sub-characteristics defined in the standard eventually leads up to a quality model called "Quality in use". The scope of this thesis is to assess a portal framework's *functionality* as defined in Chapter 3 - "Introduction" and Chapter 4 – "Definitions", and it shall specifically not assess whether the corporate portal fulfills its role as a knowledge management product or similar usage scenarios.

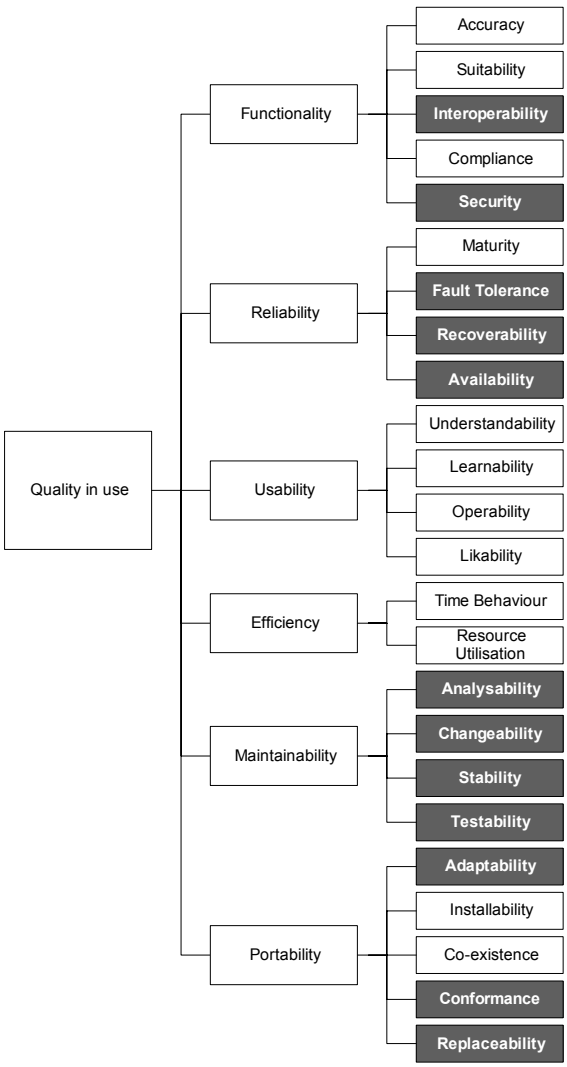


Figure 39 The ISO/IEC 9261-1 Quality Model. The grey boxes are sub-characteristics that are deemed valuable for this thesis.

Nevertheless, the standard makes use of several internal characteristics in assessing a software product's quality in use. These internal characteristics are valuable as guidelines for what to look for in the vendor analysis. Figure 39 shows the entire quality model, with the sub-characteristics deemed valuable for this thesis in a grey color. The definitions for these sub-characteristics are then listed, verbatim from the source [ISO/IEC 9261-1]. Following afterwards is some comments to these characteristics and to the selection.

14.1 Definitions

These are the definitions used from the ISO/IEC 9261-1 document.

14.1.1 Functionality

The capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.

Interoperability: The capability of the software to interact with one or more specified systems.
NOTE: Interoperability is used in place of compatibility in order to avoid possible ambiguity with replaceability.

Security: The capability of the software to prevent unintended access and resist deliberate attacks intended to gain unauthorized access to confidential information, or to make unauthorized modifications to information or to the program so as to provide the attacker with some advantage or so as to deny service to legitimate users.
NOTE: This also applies to data in transmission.

14.1.2 Reliability

The capability of the software to maintain the level of performance of the system when used under specified conditions.

Fault tolerance: The capability of the software to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
NOTE: The specified level of performance may include fail-safe capability.

Recoverability: The capability of the software to re-establish its level of performance and recover the data directly affected in the case of a failure.

Availability: The capability of the software to be in a state to perform a required function at a given point in time, under stated conditions of use.
NOTES:

1) Following a failure, a software product will sometimes be down for a certain period of time, the length of which is assessed by its recoverability
2) Externally, availability can be assessed by the proportion of total time during which the software product is in an up state.
3) Availability is therefore a combination of maturity (which governs the frequency of failure) and recoverability (which governs the length of down time following each failure).

14.1.3 Maintainability

The capability of the software to be modified.
NOTE: Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

Analysability: The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.

Changeability: The capability of the software product to enable a specified modification to be implemented.
NOTE: Implementation includes coding, designing and documenting changes.

Stability: The capability of the software to minimize unexpected effects from modifications of the software.

Testability: The capability of the software product to enable modified software to be validated.
NOTE: Values of this sub-characteristics may be altered by the modifications under consideration.

14.1.4 Portability

The capability of software to be transferred from one environment to another
NOTE: The environment may include organizational, hardware or software environment.

Adaptability: The capability of the software to be modified for different specified environments without applying actions or means other than those provided for this purpose for the software considered.
NOTES:
1) Adaptability includes the scalability of internal capacity (e.g. screen fields, tables, transaction volumes, report formats etc.).
2) Adaptability corresponds to suitability for individualization in ISO 9241-10.

Conformance: The capability of the software to adhere to standards or conventions relating to portability.

Replaceability: The capability of the software to be used in place of other specified software in the environment of that software.
NOTES:
1) Replaceability is used in place of compatibility in order to avoid possible ambiguity with interoperability.
2) Replaceability does not imply that this software is able to replace the software under consideration.
3) Replaceability may include attributes of both installability and adaptability. The concept has been introduced as a sub-characteristic of its own because of its importance.

14.2 Comments

- Several of the definitions includes the phrase "… under stated conditions of use" or "… when used under specified conditions." As this thesis does not explicitly specify usage conditions, confer the scope of the thesis, the conditions of use is set as "any applicable".

- The standard does not mention "scalability", or any similar term, as a characteristic or sub-characteristic. This is the capability of the software to scale or grow with increased usage, both in number of users and the utilization of the product from the existing users[73]. A related characteristic, "efficiency", is defined as "The capability of the software to provide the required performance relative to the amount of resources used, under stated conditions," or rather, how much "use" one get per resource. This characteristic does not encompass the need for a software product to be able to scale. If a corporation is expecting large growth in the near future, it is not interesting whether the product is efficient, but whether it

[73] Scalability for server products is most often achieved by making the server software capable of handling multiple physical server resources.

can handle increasing load. As this is an important aspect of a portal product, this is a characteristic is used in the vendor analysis.

- *Usability*, with its sub-characteristics *understandability*, *learnability*, *operability* and *likability*, is used in several of the arguments for using corporate portals. The user understands the concepts used in the portal, and it is easy to learn and operate the system. This is mostly because of the familiar user interface of a web browser and the concepts of windows loaned from familiar operating systems. Hopefully, these systems are also easy to like, since they are supposed to make one's work and duties easier. These characteristics are not relevant to the thesis' main goal, though, and are thus not included in the definitions list.

- "Vendor lock-in" is a term referring to the fact that it often is difficult to move away from a vendor when one first have started to use some of the vendor's products. As no definition clearly points to this problem, the definition "replaceability" is therefore extended to also include this software's capability to be replaced by another product.

15 Appendix B: Java 2 Enterprise Edition

The Java 2 Enterprise Edition specification [J2EE] is a set of standards set down by Sun and several working groups, consisting of many prominent vendors. The standard is especially tailored towards distributed, internet-enabled applications and other environments where resources are spread out. The application model used in the J2EE environment is shown in Figure 40.

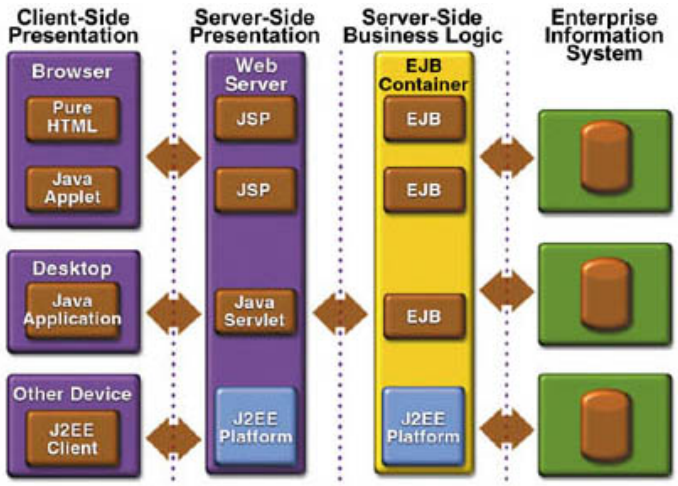


Figure 40 The J2EE application model (From java.sun.com: J2EE overview)

The full specification encompasses 13 different specification and standards, as of version 1.3. A full low-down of these specifications is beyond the scope of this thesis, but a short description of some of the concepts within the Java 2 Enterprise Edition is elucidating. Figure 41 shows an illustration of the scoping and relationships between the entities.

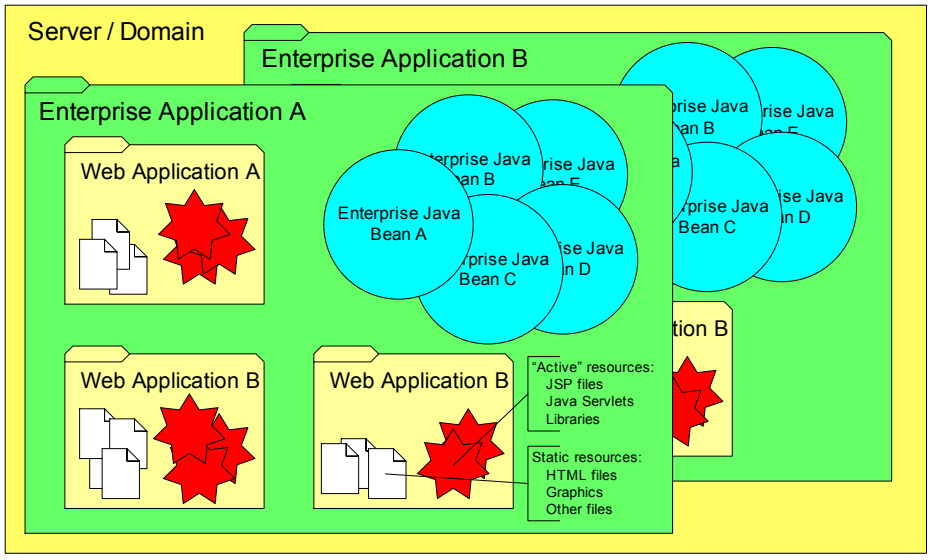


Figure 41 Scoping of entities in the J2EE specification

- A *Domain* refers to one installation of a J2EE server and all the enterprise applications deployed within it. This is a term used by BEA, and is not a part of the J2EE standard.

- An *Enterprise Application* is a set of Web Applications and Enterprise Java Beans that are deployed inside a Java 2 Enterprise Edition compliant server. The entire enterprise application may be contained within one single Java Archive, a jar-file, whose ending is ".ear".

- *Web Applications* are a set of JSP files, Servlets, Java code, Java libraries and static content like images, html files and other files. These resources are accessible beneath a specific URL. This can be either a domain, for example http://www.corporation.com/, or a path, for example http://www.corporation.com/someApplication. The web applications are run within the Web Container of the J2EE architecture. A *Web Container* (often called *Servlet Container*), may exist by itself, and does not require the full J2EE architecture set to run. A web application may be contained within one single jar file, whose ending is ".war". A web application is self-contained, and may not communicate easily with other web applications except by means of shared resources, for example, a RDBMS, or the enterprise java beans scoped within the same enterprise application.

- The *JSPs* and *Servlets* are active, in that the server will not just merely serve their contents, but will run the application code contained within them. The server provides means for the code to access the HTTP headers that the browser supplies, and means for the code to reply to the browser.

  The Java ServerPages (JSP) architecture is rather complicated. The system uses the *code-within-html* paradigm, where Java code, called *Scriptlets*, are embedded within the ordinary HTML using a special markup syntax consisting of "<%" and "%>". There is also a notion of *tags*, where HTML-looking tags within the JSP files invoke pieces of Java code. These tags are collected in *tag-libraries*, which the JSP developer imports into the JSP file by special import statements. The tags' code can later be invoked by inserting the tags and their arguments in the JSP file.

  The web container's JSP interceptor first transforms the JSP file to a Java Servlet source code file. This file is then further compiled to a Java class file using an ordinary java compiler. The resulting executable Servlet is finally invoked and run. This entire process only happens the first time a user requests the resource and when the JSP file is modified, at all other times the cached class file is invoked directly. This makes for fast code, as the files are not parsed on each request, as in the case of for example Microsoft's ASP and Apache's PHP type of code[74]. The already compiled code is merely invoked by a method call.

The objections against JSP are abundant [e.g. Stevens, 2002 and Hunter, 2000a, 2000b]. The complicated three-stage cycle of invocation makes for three entirely different error sources. The transform from JSP to Java Servlet source code must obey

[74] This is a truth with some modifications. Firstly, Java is originally also an interpreted language, where *bytecodes* are executed on a Java Virtual Machine, although this process has been supplemented by intricate *just-in-time compilers* that further compile the bytecode down to native machine code. Secondly, Apache PHP, for example, also have the possibility of being compiled to an intermediate representation, thereby accelerating the execution of code made in this language.

the JSP conventions, and may fail. Then, the resulting file may have syntactic Java errors, making the compilation fail. Then, finally, the code must run and produce HTML. This output may of course contain HTML errors. This complicated dual rewriting process produces HTML that may be quite bizarre in contrast to the original JSP file, making it problematic to find where in the JSP file the error was made. In addition, the error codes from this process is inherently difficult to decipher, especially the middle compilation step, as the coder can't control how the Java source code is generated.

In addition, JSP is supposed to divide the *Controller* and the *View* in the *Model-View-Controller* paradigm of coding[75]. The problem is that all such code-within-HTML systems[76] make it very hard to divide completely the code logic and the html, relying on the coder to be very stringent with his coding patterns. An in addition, the tag-libraries and the tag-invocations spread throughout the code will eventually make the resulting code look quite far from HTML.

An alternative to JSP is template-based systems[77]. These systems have become quite popular, and have been characterized by several authors [Stevens, 2002, DiBartolo, 2001, Hunter, 2000a, 2000b and Messerschmidt, 2001] as enforcing the MVC paradigm more rigorously than JSP.

- *Enterprise Java Beans, EJBs,* are distributable, sharable code-components. These pieces of code-logic are "living within" Java 2 Enterprise Edition servers, in the *EJB Container*. They are invoked from, for example, a web application by means of a Remote Procedure Call protocol called RMI-IIOP. The may also be invoked by other systems, for example as a part of a graphical user interface application. The enterprise java beans are of different types, called stateless, statefull, and entity beans. The statefull type is enabled by the container to automatically store their state to a persistent storage, for example a RDBMS.

An enterprise application may contain multiple web applications and multiple enterprise java beans. The web applications may communicate with these shared components. Such beans thereby provide means for sharing services between the different web applications.

- *Deployment descriptors* are XML files describing the different elements of the enterprise application. There are several different deployment descriptors for different elements of the enterprise application, each containing extensive amount of data and meta-data for the described elements. There are deployment descriptors for the entire enterprise application, each contained web application, and for each enterprise java bean. Information included is the naming of the elements, URLs used, resources needed, parameters to the

[75] J2EE BluePrints Design Patterns: Model-View-Controller
http://java.sun.com/blueprints/patterns/j2ee_patterns/model_view_controller/ [Accessed June 4, 2002]
Java Pet Store Architectural Overview (JSP MVC):
http://java.sun.com/blueprints/code/jps11/archoverview.html [Accessed June 4, 2002]
[76] PHP (http://www.php.net/), ASP (Microsoft's Active ServerPages), mod_perl (http://perl.apache.org/) and JSP are all code-within-HTML systems
[77] For example WebMacro: http://www.webmacro.org/, Apache Velocity (WebMacro inspired):
http://jakarta.apache.org/velocity/, Freemarker: http://freemarker.sourceforge.net/, Tea (Disney Corporation Open Source):
http://opensource.go.com/Tea

different entities, description of services, user role definitions, user authentication methods and so on.

- *Declarative access restrictions enable* the developers to make constraints on the different resources by querying whether a user is authenticated, and if so, whether he or she has the necessary privileges to use the resource. This is declared within the deployment descriptor for the resource.

16 Appendix C: References and Resources

16.1 Papers and Articles

Detlor, Brian (2000) The corporate portal as information infrastructure: towards a framework for portal design; International Journal of Information Management, vol.20, no.2; April 2000; p.91-101

Dias, Cláudia (2001) Corporate Portals: a literature review of a new concept in Information Management; International Journal of Information Management, vol. 21, no.4; Aug. 2001; p.269-287

DiBartolo, Vincent (2001) Freemarker: An open alternative to JSP; JavaWorld; Jan. 2001; http://www.javaworld.com/jw-01-2001/jw-0119-freemarker.html [Accessed June 4, 2002]

Eckerson, Wayne (1999) 15 Rules for Enterprise Portals; Oracle Magazine; July-August 1999; http://www.oracle.com/oramag/oracle/99-Jul/49ind.html [Accessed May 30, 2002]

Firestone, Joseph M. (1999) White Paper No. Thirteen – Defining the Enterprise Information Portal; DKMS / Executive Information Systems, Inc.; 31 July 1999; http://www.dkms.com/EPIDEF.html [Accessed April 5, 2002]

Gates, Lana (2001) The second portal wave; Application Development Trends, vol.8, no.1; Jan. 2001; p.29-32; http://www.adtmag.com/article.asp?id=2669 [Accessed June 4, 2002]

Grammer, Jeff (2000) The Enterprise Knowledge Portal; DM Review; March 2000; http://www.dmreview.com/master_sponsor.cfm?NavID=193&EdID=1940 [Accessed June 4, 2002]

Hunter, Jason (2000a) The Problems with JSP; Servlets.com; Jan. 25, 2000; http://www.servlets.com/soapbox/problems-jsp.html [Accessed June 4, 2002].

Hunter, Jason (2002b) Reactions to “The Problems with JSP”; Servlets.com; Feb 10, 2000; http://www.servlets.com/soapbox/problems-jsp-reaction.html [Accessed June 4, 2002]

Luce, Charles (2002) Three trends in enterprise information portals; ZDNet – Enterprise; March 19, 2002; http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2855580,00.html [Accessed June 5, 2002]

Margulius, David L. (2002) Plug-and-Play portlets; InfoWorld; April 26, 2002; http://www.infoworld.com/articles/fe/xml/02/04/29/020429feportaltci.xml [Accessed June 3, 2002]

Messerschmidt, Leon (2001) Take the fast track to text generation; JavaWorld; July 2001; http://www.javaworld.com/javaworld/jw-07-2001/jw-0727-templates-p2.html [Accessed June 4, 2002]

McDonough, Brian (2001) The Enterprise Information Portal; KMWorld, vol. 10, no. 10; Nov./Dec., 2001; http://www.kmworld.com/publications/magazine/index.cfm?action=readarticle&Article_ID=1124&Publication_ID=58 [Accessed May 30, 2002]

Schroeder, John (1999) Enterprise Portals: A New Busines Intelligence Paradigm; DM Review; Sept. 1999; http://www.dmreview.com/master.cfm?NavID=198&EdID=1371 [Accessed June 4, 2002]

Shilakes, Christopher C. and Tylman, Julie (1998) Enterprise Information Portals; Merrill Lynch & Co. Research Report; Nov 16, 1998; http://www.e-global.es/017/017_merril_entreprise.pdf [Accessed June 4, 2002]

Stevens, Jon and the Velocity Team (2002) You make the decision; Velocity; http://jakarta.apache.org/velocity/ymtd/ymtd.html [Accessed June 4, 2002]

Vizard, Michael (2002) Beware of portals you can't leave; InfoWorld; Vol. 24 No. 8; Feb 25, 2002; p. 8; http://www.infoworld.com/articles/op/xml/02/02/25/020225opnoise.xml [Accessed May 30, 2002]

Waters, John K. (2000) Portal Wars; Application Development Trends, vol.7, no.9; Sept. 2000; p.43-47; http://www.adtmag.com/article.asp?id=2653 [Accessed June 4, 2002]

White, Colin (2001) The evolution of the e-business portal; DM Review, vol.11, no.8; Aug. 2001; p.22-28; http://www.dmreview.com/master.cfm?NavID=193&EdID=3832 [Accessed June 4, 2002]

White, Martin (2000) Enterprise information portals; The Electronic Library, vol.18, no.5; 2000; p.354-362;

16.2 Other Resources

IntranetFocus (2002) Portal Vendors; author: White, Martin; http://www.intranetfocus.com/Portals/Portal_Vendors/portal_vendors.html [Accessed March 14, 2002]

IBM Web Services Security (IBM-WS-S) Web Services Security; IBM; http://www.ibm.com/developerworks/library/ws-secure/ [Accessed June 4, 2002]

IBM WSRP (IBM-WSRP) Web Services for Remote Portals; IBM; http://www.ibm.com/developerworks/webservices/library/ws-wsrp/ [Accessed June 4, 2002]

IBM WSXL (IBM-WSXL) Web Services Experience Language; IBM; http://www.ibm.com/developerworks/webservices/library/ws-wsxl/ [Accessed June 4, 2002]

IBM WSRP presentation (IBM-WSRP-WSRP) Web Services for Remote Portals; IBM; http://www.oasis-open.org/committees/wsrp/presentations/wsrp_high_level.ppt [Accessed June 4, 2002]

Microsoft Web Services Security (MS-WS-S) XML Web Services Security; Microsoft; http://msdn.microsoft.com/ws-security/ [Accessed June 4, 2002]

Java 2 Enterprise Edition (J2EE) Java 2 Platform, Enterprise Edition; Sun; http://java.sun.com/j2ee [Accessed June 4, 2002]

Java Specification Request 16 (JSR 16) J2EE Connector Architecture Specification; JCP; Final Release Sept. 24, 2001; Web site: http://java.sun.com/j2ee/connector/ [Accessed May 25, 2002]

Java Specification Request 155 (JSR 155) JSR 155 – Web Services Security Assertions; JCP; Initiated Oct 29, 2001; http://www.jcp.org/jsr/detail/155.jsp [Accessed June 4, 2002]

Java Specification Request 162 (JSR 162) JSR 162 – Portlet API; JCP; Review ballot Jan. 22, 2002, withdrawn Jan 20, 2002; http://jcp.org/jsr/detail/162.jsp [Accessed June 4, 2002]

Java Specification Request 167 (JSR 167) JSR 167 – Java™ Portlet Specification; JCP; Review ballot Jan 28, 2002, withdrawn Jan 20, 2002; http://jcp.org/jsr/detail/167.jsp [Accessed June 4, 2002]

Java Specification Request 168 (JSR 168) JSR 168 – Portlet Specification; JCP; Initiated Feb 11, 2002; http://jcp.org/jsr/detail/168.jsp [Accessed April 22, 2002]

Owendo (2002) The Lowdown on Enterprise Information Portals (EIP); http://www.owendo.com/technocorner/resources/rd/rd_1.html [Accessed Jan. 25, 2002]

PortalsCommunity (2002) PortalsCommunity.com Fundamentals; author: Unitas Corporation; http://www.PortalsCommunity.com/library/fundamentals.cfm [Accessed Feb. 28, 2002]

Security Assertion Markup Language (SAML) XML-Based Security Services TC (SSTC) Security Assertion Markup Language; Oasis; http://www.oasis-open.org/committees/security/ [Accessed June 4, 2002]

Standard (ISO/IEC 9126-1) Information Technology – Software quality characteristics and metrics; Committee Draft, dated Jan 29, 2001; Final version, Published International Standard of June 21, 2001, available at http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=22749 [Accessed May 24, 2002]

Web Services Component Model (WSCM) OASIS Web Services Component Model (WSCM) TC ; Oasis; http://www.oasis-open.org/committees/wscm/ [Accessed June 4, 2002] (superseded by WSIA)

Web Services Interactive Applications (WSIA) OASIS Web Services for Interactive Applications TC; Oasis; http://www.oasis-open.org/committees/wsia/ [Accessed June 4, 2002] (formerly WSCM)

Web Services for Remote Portals (WSRP) OASIS Web Services for Remote Portals (WSRP) TC; Oasis; http://www.oasis-open.org/committees/wsrp/ [Accessed June 4, 2002]

Web Services User Interface (WSUI) WSUI: Delivering Application as Web Services; Initiated by Epicentric; http://www.wsui.org/ [Accessed June 4, 2002]

16.3 Vendor specific Documents and White Papers

16.3.1 Apache Jetspeed

Apache Jetspeed's documentation is found on the Jetspeed website: http://jakarta.apache.org/jetspeed/

[Apache A] Portlet Howto http://www.bluesunrise.com/jetspeed-docs/PortletHowTo.htm [Accessed May 12, 2002]

16.3.2 BEA

BEA WebLogic Portal documentation is found on BEA's comprehensive documentation site: http://edocs.bea.com/, portal specific: http://edocs.bea.com/wlp/docs40/index.htm. No registration is necessary.

[BEA A] WebLogic Portal (Datasheet); no date;
file: portal_ds.pdf

[BEA B] BEA WebLogic Portal – Architectural Overview – Version 4.0; October 2001; http://edocs.bea.com/wlp/docs40/pdf/architec.pdf [Accessed June 4, 2002]

[BEA C] BEA WebLogic Portal 4.0 – Getting Started with Portals and Portlets – Version 4.0; March 2002; http://edocs.bea.com/wlp/docs40/pdf/portal.pdf [Accessed June 4, 2002]

[BEA D] WebLogic Portal – Personalization and Interaction Management (Datasheet); no date;
file: wlp_pers_intmgmt_ds.pfd

[BEA E] WebLogic Portal – Integration Services (Datasheet); no date;
file: wlp_int_svcs_ds.pdf

16.3.3 Epicentric

Epicentric's documents can be found at the Epicentric Resource Library: http://www.epicentric.com/solutions/resource_library.jsp. Registration is necessary to gain access to most white papers.

[Epi A] Scalability and Capacity Planning Overview; no date;
file: epi_wp_EFSScalability.pdf

[Epi B] Epicentric Foundation Server 4.0 – What's New; no date;
file: EFS_4_0_whatsnew.pdf

[Epi C] Enterprise Portal Management Systems – The Next Generation Model for Business Portals; no date;
file: epi_whitepaper_EPMS.pdf

[Epi D] Epicentric – Foundation Builder; no date;
file: efb.pdf

16.3.4 IBM

IBM's portal specific documentation is available at portal library: http://www.ibm.com/software/webservers/portal/library/

[IBM A] Portlet Development Guide – Working with the Portlet API 1.1; April 2, 2002;
file: V41PortletDevelopmentGuide.pdf

[IBM B] Portlet Migration Guide – Second Edition; April 22, 2002;
file: V41PortletMigrationGuide.pdf

[IBM C] Portlet API first draft; author Stephan Hesmer, Stefan Hepper, Thomas Schaeck from IBM; added to CVS repository for Jetspeed December 25, 2001; http://cvs.apache.org/viewcvs.cgi/*checkout*/jakarta-jetspeed/proposals/portletAPI/PortletAPIDraft.pdf?rev=1.1&content-type=application/pdf [Accessed June 4, 2002]

16.3.5 Oracle

Oracle's documentation is available from the Oracle Technology Network at http://otn.oracle.com/. Some documents require registration for access.

[Oracle A] Oracle 9iAS – An Oracle White Paper; January 2002;
file: 9ias_twp.pdf

[Oracle B] J2EE and Microsoft .NET – An Oracle White Paper; April 2002;
file: J2EEandNET_wp.pdf

[Oracle C] Oracle9iAS Portal Release 2 – Technical Overview – An Oracle White Paper; April 2002;
file: PORTAL_V2_TWP_FINAL.pdf

[Oracle D] Oracle9i Application Server – Java2 Enterprise Edition Facilities and Design Considerations; May 2001;
file: OC4J_TWP.pdf

[Oracle E] Oracle9i Application Server Release 2 – Data Sheet; December 2001;
file: 9iappserver_ds.pdf

16.3.6 Plumtree

Most of the Plumtree documents used are available at the following web site:
http://www.plumtree.com/webforms/MoreInfoFormTemplate.asp?formkey=14
However, registration is required to gain access. In addition, the Plumtree demo portal, at http://portal.plumtree.com/, contains all documentation.

[Plumtree A] The Corporate Portal Market in 2002; Feb. 18, 2002; file:
Corporate_Portal_Survey_White_Paper_February2002.pdf

[Plumtree B] Enterprise Class Gadget Web Services and Frameworks; no date;
file: Framework_Whitepaper.pdf

[Plumtree C] The Plumtree Corporate Portal Technical White Paper; no date;
file: Plumtree_4_5TechnicalWP.pdf

[Plumtree D] The Internet Architecture of a Corporate Portal; no date;
file: Internet_Architecture_Whitepaper.pdf

[Plumtree E] Plumtree Gadget Web Services – Web-Enable Your Entire Business; no date

[Plumtree F] Plumtree Technology and Channel Partners; no date

[Plumtree G] The Plumtree Difference; no date