36 Portals

If you have not read the book (*Performance Assurance for IT Systems*) check the introduction to "More Tasters" on the web site http://www.b.king.dsl.pipex.com/ to understand the scope and objectives of tasters.

Portal technology is integrated with current web and middle tier technologies, and as such it makes use of the facilities that come with them. This taster concentrates on Portal specifics and it does not repeat the relevant detail with respect to these underlying facilities. It is recommended that the appropriate tasters in the book are read if an overall understanding is required, in particular: *Network Basics, Server Load Balancing, Web Server and Cache Server, LDAP Server, and Modern Server-Side Development Technologies.*

36.1 Starting Terminology

JSR 168 is a standard API for creating portlets. It has been developed via JCP (the Java Community Process) with contributions from 20 major IT vendors.

A **Portlet** is a web-based component that will process requests and generate content. The end-user sees a portlet as a specialized content area that occupies a small window in the portal page.

WSRP (Web Services for Remote Portlets) is one of a number of standards from OASIS (Organisation for the Advancement of Structured Information Standards), on which many of the major IT vendors are represented. WSRP facilitates not only the reuse of a remote back-end service but also the reuse of the user interface aspects of that service, both under the umbrella of web services

36.2 Technology Outline

36.2.1 What Is A Portal?

There are probably as many views of precisely what a portal is as there are people who are asked for an opinion. From a sizing and performance perspective there are (simplistically) two classes of portal:

- A "thin portal". I define this type of portal as one where there is little interference with the user's session and hence the processing overheads are relatively modest. Examples include: a simple routing capability where the portal contains the necessary links to allow the user to get to the system / data that he requires; or a single sign-on system where the portal may play no further role once sign-on has been complete and initial routing has taken place

- a "fat portal" is defined as one where the portal controls the user interface, typically dividing a web page into a number of discrete areas, sometimes called controls, each of which is populated by data from different sources / applications that are invoked by the portal. Portal products fit this description. Needless to say, the processing overheads are considerably greater than the thin portal. This taster will focus on the fat portal.

Many vendors will produce a diagram similar to that shown in Figure 36-1 to describe the basic features of their products. An on-line learning environment is shown, but merely as an example.

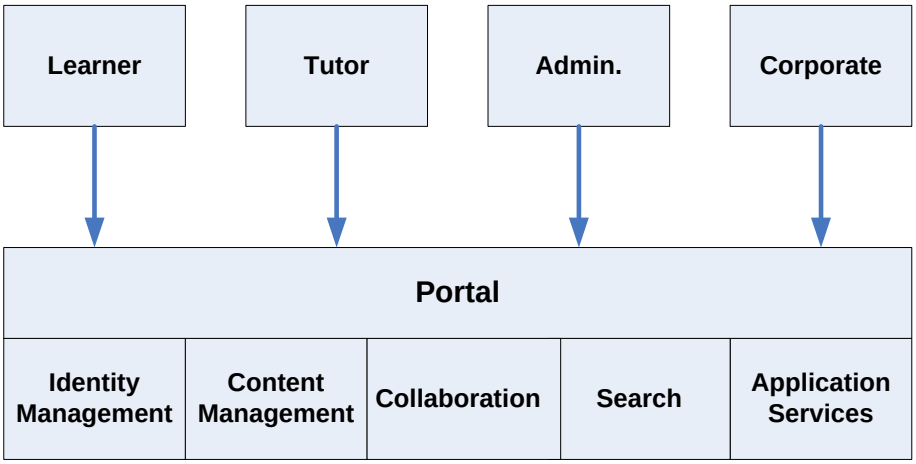

Figure 36-1 High Level Overview of Portal Users and Features

- Identity Management will typically include authentication, possibly a single sign-on mechanism, and either highly or loosely integrated access controls that define a user's roles and privileges. An LDAP server will be used, along with either an integrated or external access control system.

- Search may be limited in scope to items that are locally indexed, although some more sophisticated search features may allow federated searches over multiple external data sources.

- Content Management can provide a centralised method of storing and publishing all content that is used across the organisation, making use of workflow and version control techniques, as required.

- Collaboration typically includes facilities to track projects, share documents, exchange ideas and send messages / emails.

- The features above are sometimes referred to as Foundation Services. Application Services indicate the ability to invoke services from other bespoke or product-based applications. Not all portal products offer this facility.

36.2.2 Architectural Positioning

Portal systems require many of the infrastructure facilities that are available in web and other multi-tier based applications, e.g. authentication, load balancing, clustering, scalability, high availability, *et al*. It is sensible to make use of existing approaches to these topics, rather than attempting to reinvent the wheel. For this reason, the favoured positioning for a portal server is within the middle tier, usually under the control of an application server, where it can readily make use of these infrastructure components. This positioning is leading application server vendors to include a portal server offering as part of their portfolio.

Figure 36-2 shows this positioning plus the connection to Identity Management and the Portal repository. The architecture to support authentication can vary from system to system. For example, a reverse proxy may be employed that communicates directly with Identity Management. Identity Management will include an LDAP server, while some products will additionally support external directories that may be used elsewhere in the organisation via some form of "directory integration". Every portal server will have a repository, sometimes called a configuration database, to hold metadata; this will include page definitions and personalisation details for each user. The repository is typically housed in a relational database that is accessed via ODBC or JDBC drivers.

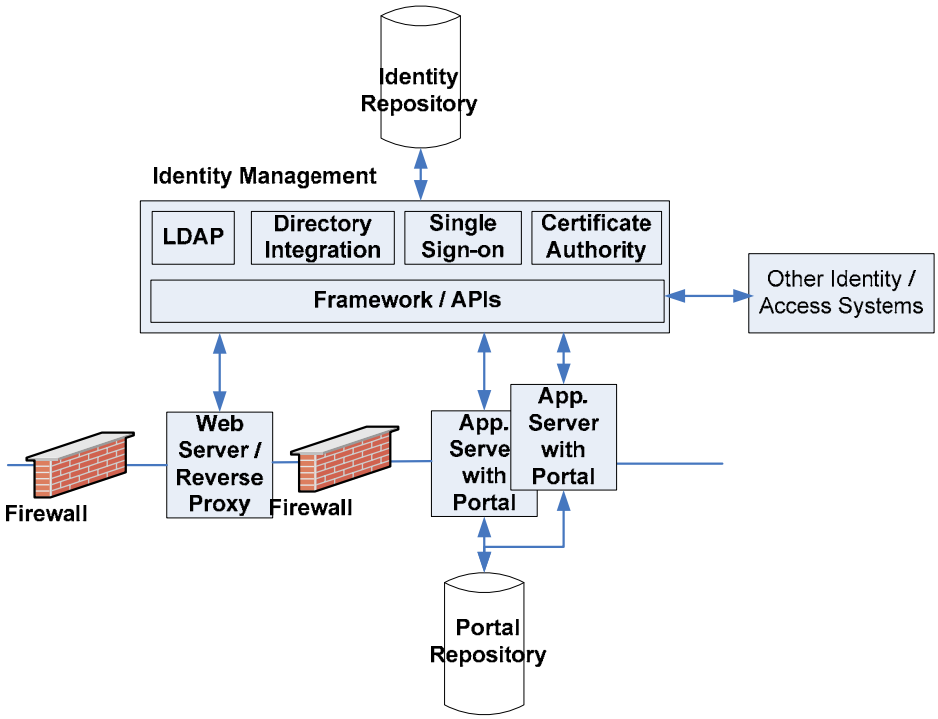

Figure 36-2 Portal Positioning / Interfacing with Identity Management / Portal Repository

36.2.3 Portal Server Message Processing

The outline processing of a portal server when handling a message request is illustrated in Figure 36-3. Before explaining the diagram it should be noted that where multiple portal servers are used to provide scalability, sticky sessions are used, i.e. a user is allocated to a given application server when she logs on and remains on that server for the duration of her session.

The first step when handling a message is to retrieve the necessary page definition from the portal repository and parse it. It is advantageous from a performance perspective if the parsed version can then be cached in memory on the middle tier for future use, as retrieving and parsing can be significant overheads. Each control (discrete area of the portal page) then has to be populated with the necessary data. This is typically achieved by the portal server engine invoking a portlet, i.e. a web-based component.

As an aside, JSR 168 specifies a standard API for portlets. Although it is meant to be vendor-independent, it shows its Java heritage by treating portlets like servlets. Microsoft has, possibly unsurprisingly, failed to recognise this standard. Communication between the portal

server engine and the portlet is typically achieved by the use of SOAP-based messages that are transported via HTTP. On the theme of standards, WSRP (Web Services for Remote Portlets) not only supports the reuse of a remote back-end service but also the reuse of the user interface aspects of that service, both under the umbrella of web services. It appears to have been broadly adopted by vendors, including Microsoft.

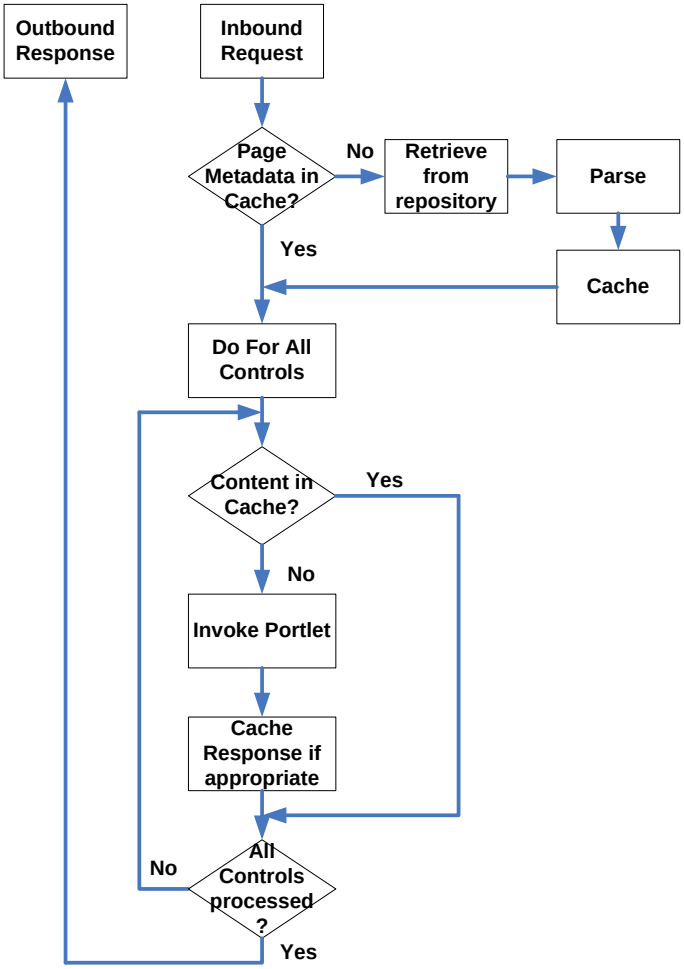

Figure 36-3 Overview of Message Processing by Portal Server

Returning to the portlet invocation, as a portal page may contain many controls, any reasonable portal server engine should have the ability to invoke multiple portlets in parallel in an effort to reduce response times. The retrieved content for each control is placed on the page. Once again, it will be advantageous if the content for each control can be cached on the middle tier to avoid extraneous portlet invocations. Note that some products may only support local portlets, i.e. components within the same environment, whereas other products may also support remote portlets.

Although Figure 36-3 only mentions the invocation of portlets, a portal server may invoke services by other means, e.g. the interface may be via TCP/IP, although such facilities may simply be to support other software that does not conform to portlet standards and therefore they may not be openly available for use by the customer.

36.3 Performance

Page Complexity. Busy portal pages that contain many controls and involve the invocation of many portlets will naturally increase the use of hardware resources and may well result in poor response times. It is probably wiser to take a pragmatic approach here when designing the portal. Try to limit the number of controls, possibly by creating some form of hierarchy where there is a main portal page and a number of sub-portal pages.

Caching. As mentioned previously, the ability to cache system data (e.g. page definitions) and content (ideally at the portlet level) can significantly improve performance. Where entire portal pages are mainly static once they have been constructed, the use of a standard web server memory cache or a web cache server can aid performance. See the *Web Server and Cache Server* taster for further information.

Large Rate of Log-ins. One key factor in any sizing is the ratio of logins to transactions. The smaller the ratio, i.e. the greater the number of logins, the greater the processing requirement will be. This will naturally also affect the Identity Management element of the system. Seek guidance from the vendor in this area.

Sizing Guidelines. If using vendor rule of thumb sizing guidelines beware, as mentioned in the observations section, that they may be based on unrepresentative benchmarks. It may be necessary to increase their figures in line with an assessment of how the complexity of your proposed portal compares with their benchmark. If there are no hard sizing metrics available use 50-60% of the application server metrics that you will find in the example hardware sizing in chapter 5, *Hardware Sizing: The Crystal Ball Gazing Act.* Similarly, use 25-30% of the database server figures for the portal repository. As I always say: use these metrics with extreme care. Performance can, and does, vary from product to product.

Sizing The Services. It is important to remember that the sizing metrics that are mentioned above only apply to the portal server and its repository. In addition, each of the products that are used in conjunction with the portal server, e.g. collaboration or custom application services must be sized separately.

General tuning. A portal server typically runs under an application server and therefore the general tuning guidelines for that application server should be followed

Deployment Options. While it may be possible to deploy a portal server, repository and all other related software on a single hardware server, this approach is not recommended except for a very small system, a modest production pilot, or an acceptance testing system. The first step in a multi-server deployment is to separate out the Identity Management and repository processing and place them on their own servers. In the case of Identity Management, the directory server(s) and any access control servers may well be company-wide services that require their own servers anyway. The portal repository can share a database server with other systems. However, as vendors are keen to state, maximum performance can be obtained by putting it on a dedicated server – although this comment is more applicable to larger portals. Other components that may warrant dedicated servers include: indexing for searches, particularly if this is done at peak times; portal administration tasks; plus collaboration and content management; particularly if they use different technologies to the portal server. The portal server itself can usually be scaled up by deploying it across multiple application servers.

36.4 General Observations

Benchmarks. In contrast with many other areas of IT technology, there are very few portal benchmarks whose results are available in the public domain. Nobody has so far attempted to specify a standard portal benchmark. This lack of activity may be due to the effort that is required to set up a representative benchmark, coupled with the limited size of many portals thus far and therefore the size of the market. While the reduced opportunities for vendors to engage in competition-bashing are welcome, it does make sizing somewhat more problematic without resorting to in-house benchmarking.

What benchmarks I have seen tend to employ the usual "tricks of the trade" to ensure good performance, e.g. predominantly read-only transactions, only a modest number of portlets invoked (typically 3 or less), and a very low user hit rate (which is OK *per se*) but this is often translated into the ability to support very large user populations (one million seems to be a popular number!).

Difficulties in Establishing Volumetrics. The term "Portal" is one of the current buzz words in the IT industry. In many instances, it is unfortunately the case of a technology seeking a solution. I have seen a number of papers that have been written for clients, where the authors have been both in-house staff and external consultants, which use the word "Portal" as their starting point rather than the term "business case". Several proposed systems have unsurprisingly been cancelled when the lack of a business case has been belatedly exposed. It follows that the availability of volumetrics on which to base any hardware sizing can be extremely difficult to find; and in a number of large projects they have been non-existent.

The first question is to ask is "is there a good work-related reason why a user should use the portal on a daily basis?" If the answer is yes, e.g. project-related data is indispensable, it should be possible to get a reasonable handle on the classes of user and their respective frequency of usage. This type of portal is likely to be significantly larger than one where usage is *ad hoc*.

If the portal is more speculative, e.g. making information available that is considered potentially useful, the chances are that actual usage may well be significantly lower than any project or management expectation. One possible method of deriving volumetrics here is to:

- start by classifying the types of user. For example, a rudimentary split may consist of internal users (employees) and external users (customers or potential customers).

- The next step is to decide what a typical session profile may look like: the internal user's session may consist of 30 hits (clicks); while the external user may be only 50% of this figure.

- The third step is to estimate the frequency of the sessions: the internal user may use the system once per day while the external user may only use the system once per month. If it is assumed that there are 500 internal users and 2000 external users, the average hit rate (over an 8 hour day) will be less than one per second. The peak rate, even if it was 3-4 times the average rate, will still only be in the range of 2-3 transactions per second.

While I am not for one moment suggesting that these session profiles and frequency rates are used parrot-fashion (rather use the approach as framework and employ your own numbers), they are not wildly inaccurate for a system where usage is likely to be unknown or ad hoc.

The level of peak user concurrency in the above example is ~2%. While this may sound low, there are portals where the concurrency rate is less than 1%. The bottom line is that in this type of scenario, it is eminently reasonable to start with a small system, so long as it is possible to grow quickly and fairly painlessly if there is a true demand. Read chapter 5, *Hardware Sizing: The Crystal Ball Gazing Act* for further discussions on deriving volumetrics for hardware sizing calculations.

36.5 Further Reading

There appears to be little useful material on this subject in the public domain. I can only assume that there is no significant interest in this area in academia. Freely available papers from portal vendors tend to be limited to descriptions of features, coupled with comments on how to approach the design of a portal. Administration guides, if you can get hold of copies, are slightly more useful in describing the architecture of the product, providing some sizing guidelines and discussing hardware deployment options. Having said that, the only satisfactory documentation that I have seen thus far is for Oracle's AS Portal.

Web sites include:

http://www.oracle.com/technology/index.html Oracle's Technology Network which includes useful information on their AS Portal product .

http://www.oasis-open.org/ the OASIS site (WSRP standard).

http://jcp.org/ the JCP web site which includes information on JSR 168, and indeed all JSRs.

