Portal Essentials

Chapter 1:
Introduction

(draft)

Gartner Group refers to "portal" as the "most abused term in IT." Perhaps this is one reason few books tackle the subject of portals. One widely employed, though loose, definition of a portal is "any web page that a user can log in to", but this falls far short of defining what makes a portal a portal. Another suggests a portal is a portal because of its ability to combine content from multiple sources, but this too is only part of the portal story. This chapter begins the rehabilitation of the term "portal" by offering a solid definition of a portal that addresses the following questions:

- What makes a portal a portal?
- What is personalization? How does it differ from customization?
- What is a channel? What are data, Web, application, information cameos?
- How do portals use content? What is content aggregation?
- What is middleware? How do middleware and directories work with portals?
- How might portals use Web services? What is a portlet?
- What XML specifications apply to portal development?

Introduction to Portals

A portal aggregates content and services. Aggregation is simply combining content from multiple locations and presenting it in a common context. An example of aggregation might be a newspaper web page. While a physical newspaper consists of groupings of pages that constitute sections, a newspaper web page is more typically an aggregation of headlines from all the newspaper sections, grouped by section.
But every visitor to this newspaper site sees the same collection of headlines; there is no user-specific intelligence that affects the aggregation or resulting page. So aggregation is only part of the portal story.

Portals require users to log in or identify themselves to the application. This simple authentication step enables portals to do much more than generic content aggregation. The functions that portals provide can be grouped into four categories: personalization, customization, integration, and support for online communities. Since portal users are not anonymous, portals can personalize the user's experience based on what they know about the user. A portal also provides users with mechanisms to customize their

experience, and stores that information. A portal is a platform for integration with enterprise information systems and applications. Finally, portals provide support for online communities. So any Internet-accessible application that aggregates content and services, provides customization and personalization, is integrated with business systems and Web applications, provides support for online communities, and does all these things for known users, is a portal.

A portal personalizes the user's experience

Jacob Nielsen, a noted usability expert, says personalization "is driven by the computer which tries to serve up individualized pages to the user based on some form of model of that user's needs." In order for personalization to occur, a user must identify himself to a portal when he first accesses it. This allows the portal to retrieve information that it has access to about the user and to make decisions about what they are presented with, even if they have never logged in before. In its simplest form, personalization might involve determining what the user's primary role is within an organization (manager, employee, customer, CEO), and building a page of content by using a template that has been predefined for that class of user. This determination might utilize a database containing a local copy of core data about an individual, or middleware that allows the portal to tap into a directory or ERP system. This directory or database affected view of the portal is personalization.

A portal allows the user to customize their experience

A portal can also contain data that the user provides directly. Typically, this would include customization preferences, that is data that the user supplies through a customization interface that a changes the portal's presentation. Some content and services might be locked in place, to reduce support issues and provide some uniformity to users and some assurance to the sponsoring organization that its goals are being served by the portal. But other aspects of the presentation can and should be free form and user-configurable. This user-affected view of the portal is customization.

A portal is a platform for integration

A portal's role in integration can be extensive. Some integration involves gaining access to information about users. If a portal is integrated with an enterprise information system (EIS), then it can apply a set of rules to define the user's experience and make authorization determinations, without requiring that the user fill out lengthy forms. Integration can also involve providing single sign-on access to other existing Web applications. Single sign-on provides convenience, and it can even improve security of a Web infrastructure, if properly implemented. But relying too heavily on a portal for integration can also introduce problems as we will see in a later chapter.

A portal provides a support framework for online communities

Online communities can play a variety of roles within an organization. This might include virtual staff meetings, collaborative document authoring, and online course discussions. Portal support for online communities can range from discussion forums and collaborative content development, to instant messaging services, group announcements, and calendar channels.

Why Portals?

Users and organizations have differing but overlapping expectations of portals. Users are most interested in quality of information and services, persistent services that can be accessed from any location, and the ability to customize content. Organizations are looking for a way to integrate legacy applications, vendor supplied products and locally developed solutions in a secure environment with a consistent application development interface. Let's examine these expectations in a bit more detail.

What users want from portals

Users have flocked to information-oriented portals such as Yahoo for years. Some came to find information. Human classification of information still edges out machine classification, because humans are better at making determinations with regard to quality, and better at considering other factors such as the author and sponsor of content, thus recognizing things that might affect the perceived quality of information such as a vendor's subtle marketing pitch embedded in a technology white paper.

Others are attracted based upon the availability of one specific service. Hotmail has attracted millions of users, because it is free and sign up is easy. It appeared on the Web at a time when many enterprises were still grappling with technology that would make their own enterprise mail system available on the Web. A Hotmail user might graduate from college, or change jobs, but their Hotmail account does not change.

Still other users were attracted by the availability of customizable Web pages. My Yahoo allows users to select and arrange a variety of Web content. One a user customizes their "My Yahoo" page, they see the same set of channels arranged in the same way every time they visit the site. But each channel refreshes itself automatically, as needed, so news headlines and stock valuations are current. The customizable dashboard-style Web page has been found to be much stickier than non-customizable equivalents. In fact, one study indicated that users were ten times more likely to return to a Web page that they could customize.

What organizations want from portals

Many organizations are slow to adopt portals. The first portal-like solution that appears is usually a structured directory of information. Informational portals usually incorporate pointers to Web applications as well, but provide no integration with them. If an application requires authentication, the information portal merely drops the user off at the login screen. Only when an organization builds, deploys or purchases an enterprise solution do they begin to explore integration opportunities, typically starting with communication tools such as e-mail and group calendaring solutions.

Organizations that invest in enterprise portal solutions want to lower the cost of owning, integrating and developing applications. They are also concerned with providing better customer service, and self-service solutions whenever possible. Portals can be used to leverage investments in knowledge and content management and provide a framework in which users can access information and create new knowledge collaboratively.

Portals also allow organizations to leverage their investment in business and customer relationship management solutions. Whether inward facing (intranet) or outward facing, a portal can use data that already resides within the business management system to personalize a user's experience and manage authorization to Web applications, thus allowing portal integrated applications to forego explicitly requiring users to login again (single sign-on). Portals can aggregate content and present it through channels, making optimal use of screen real estate and user attention. Portals can provide customers with self-service solutions such as product watch lists, personalized views of catalogs, synchronous or asynchronous customer support services, and management of payment and shipping information.

Portals can provide end-to-end integration of new and existing legacy systems. Legacy systems can share data and be integrated with portals though platform and language-neutral messaging systems and application interfaces such as those specified by the Web services suite of standards. The portal can present discrete services as individual channels with minimal re-engineering of the underlying application, depending on how users will be accessing the portal and how the application was originally implemented. These channels can be presented alongside other content and service-oriented channels. Personalization will determine who can see and access which channels. Then users can then further tailor their experience through the portal framework's customization mechanism.

Channels, Channels Everywhere

Portals deal in the currency of channels. A channel is a highly focused set of information or a discrete mechanism for performing a very specific task. Developers have the luxury of "stretching their legs" in the context of a Web application, but in a portal screen real estate and competition for attention are considerations that are less apparent or absent in other environments. On the plus side a channel lives in an environment where its interactions with the underlying framework are well known. It also has privileged access to data about the user and the tasks they may have performed than a stand-alone application might have. These characteristics reduce the cost of developing applications while simultaneously enabling new types of services and a new context for delivery over which the user has some control.

Channel types can be further subdivided into categories that represent the level of interaction the user has with their content, and the level of integration between the channel and the underlying portal framework. Some make a further distinction referring to the presence of some content as a "cameo." Howard Strauss proposes that there are three types of cameos that can appear within portals: Web, data, and application.

A Web cameo is in a sense a legacy integration model. Existing Web content that was not designed specifically for delivery through a portal but that has been repurposed and delivered within a portal channel is a Web cameo. An example of this is a set of news headlines or a weather forecast. These might be directly incorporated into the portal if they have a portal-friendly design that is compact and makes careful and efficient use of screen real estate. Or it might be content that is "screen-scraped" from another site. Copyright issues aside (and they are numerous), screen scraping is generally a very poor substitute for building a true portal channel that has access to needed data, and is designed specifically to present the data within a portal. This is where Strauss's application and data cameo concepts really shine.

A data cameo is a view into some live data that is directly relevant to the individual who is accessing it. This might be output of a stored query, a summary of a report, or the status of a transaction. In essence, a data cameo is a report channel. It is a succinct, personalized and/or customizable view into some data that the portal does not own. A data cameo does not have to be interactive but it is always dynamically generated either at request time, or prior to it by some application that examines data and summarizes it. A data cameo is in a sense ephemeral. It contains little persistent content. So it is very important to incorporate metadata about a data cameo into its definition so that users can find it.

An application cameo is interactive. It represents a discrete task that a user can perform. It has a user interface, usually forms-based, that allows a user to input or manipulate some data. An application cameo is not usually an entire application; instead it is a subset of functionality that an application might provide. It usually represents a frequently performed task. Here again the channel itself contains little information that tells the user what it is about. So it is important to incorporate metadata into the definition of an application cameo, so that it can be located.

Figure 1-2
At left is a Web cameo that simply displays a portion of a Web page within a channel. On the right is a data cameo that displays data specifically requested for the individual currently viewing the channel.

We also add an additional cameo type to the set described by Strauss: the informational cameo. Superficially this may seem to overlap with the data cameo but there are a number of characteristics that make information cameos distinct.

An information cameo is not personalized to the individual user and it is rarely customizable. It may be dynamically generated but most likely not at the time the user requests it, because the content is not personalized for the end-user. It is typically implemented using richer content-specifications suitable for data such as news headlines or tables of contents, most commonly RSS (Rich or RDF Site Summary). Metadata embedded within an information channel typically describes its current content as well as its overall purpose, whereas a data cameo would rarely if ever contain metadata that related in any way to its current contents, except in very general terms (e.g. XYZ Bank savings account balance). An information channel can contain a search option that

allows users to discover additional content, such as past versions of the channel, or articles to which the channel points. Finally, it is usually syndicated content and often contains data that defines its own "life cycle." Syndication metadata can establish expiration and refresh cycles for an information channel. This allows the portal framework to determine when the content needs to be updated, so it can refresh and cache the content.

Figure 1-1
At left, an application channel through which the user could search a database. On the right, an information channel containing headlines from BBC News.

The cameo concept works well as a way to categorize channels and their level of interactivity and customizability. It can be awkward to use however, since it was intended to describe a piece of content within a portal channel rather than a channel, and this is a distinction that is rarely of use to developers. What people tend to think of as a channel and what Strauss calls a cameo are often functionally identical. Occasionally, there are channels that combine content from different sources and it is possible in those cases for the content to represent two different cameo types. Usually though such chaining takes place behind the scenes and only the result is presented to the end user. But in most cases, it is equally acceptable to refer to a channel as an application cameo, or to say that a channel contains an application cameo. Only compound channels contain multiple cameos.

Portals and Content

Structured content and metadata can be used by portals as well as by end users. Portals can reference metadata and content elements to perform personalization, build searchable indexes, and recommend content. They can also summarize content not specifically designed for delivery through the portal, and market content and services through channels. A portal can filter and proactively seek content on a user's behalf by matching user demographic data and user preferences to content metadata. Metadata crosswalks allow portals to map between different elements in different types of metadata, when there is an intersection between them. Taxonomies and control vocabularies can be used to make comparisons between the contents of metadata elements describing documents.

Portals and Middleware

Messaging, directories, authentication and authorization services are all fundamental classes of middleware that portals use. IBM once described middleware as the "the sweet, nougaty center of infrastructure". Middleware is any one of a number of application types, application design practices, data storage and retrieval systems, and data transfer technologies that enable e-commerce, e-communications, authentication and other services. Middleware tends not to have a user interface, but rather comprises the logic layer between other applications and systems.

A portal has the potential to deliver timely information through announcements, workflows, and other mechanisms and to achieve a measure of reliability in delivering that content that only e-mail and instant messaging provide today. Users tend not to be

especially loyal to or consistent in their visits to Web sites. Only news and weather sites generally attract anywhere near the attention users pay to their e-mail inbox, even if it is full of spam. Portals combine interactivity with personalized, customizable access to information and applications with one goal being attracting repeat visits from users.

The stickiness of e-mail has a lot to do with its integration with directories and middleware. E-mail servers rely on the contents of directories to determine who is authorized to use the service, and how to route incoming e-mail. Users can login to and check e-mail, and find and exchange e-mail with other users thanks to directories and middleware. Directory enabling a portal provides many of similar benefits to portal users. Users can login to portals, see personalized content, and find and communicate or collaborate with other users because of directories and middleware. Portals often interact with and exchange data with various types of middleware through various XML content and protocol specifications.

Portlets

Portals are containers for applications. But they would do little to reduce the developer's burden if they were merely indistinct buckets. Standard programming interfaces allow developers to reuse code and focus more on the core mission of an application. Portlets represent a standard for implementing applications within portals. The concept is similar to the use of JSP and servlets to simplify Web application development. The portlet API defines how a portal application interacts with the underlying framework and how it responds to input, making it possible for portal application and data channels to be portable between Java-based portal frameworks that support the portlet API.

Portlets are an excellent compliment to Web services. Web services is, in a sense an "interface to everything." Most languages support Web services so many applications can be made into Web services. While Web services is not a portal-specific specification, it has characteristics that make it an enabling technology for portals, including a standard mechanism for discovering services (UDDI) and learning about their interface (WSDL), and communicating with them (SOAP). The portlet model provides an excellent general-purpose channel development framework in which to implement consumers of Web services to make them portal channels.

Another standard further extends the relationship between portals and Web services. The Web Services for Remote Portals (WSRP) specification provides a framework for Web service providers who want to design especially for portals. The goal is to allow Web services to be published within portals without the need for any additional consumer-side coding. In other words, WSRP-compliant Web services are "plug-and-play" solutions that can be dropped into WSRP-compliant portal frameworks without the need for additional service-specific coding on the portal side. The specification also addresses user interface rendering rules with which the service must comply in order to be displayable and customizable within a portal. A portal can also discover these types of services in UDDI directories that provide access to WSRP-compliant Web services.

Chapter ?? will provide a more complete introduction to portlets and WSRP.

Portals: The XML Platform

Portals use XML extensively. As we've seen portal channels are implemented with XML formats such as RSS or XHTML and rendered by applying XSL to generate device or browser specific output. User data is often supplied through XML-encoded extracts or XML-tagged DSML messages. Document metadata is wrapped with XML. Classification information that facilitates personalization can be retrieved from XFML-encoded taxonomies, ontologies, and control vocabularies distributed around the Web. Portals can be consumers and aggregators of Web services. A data channel might call one or several Web services in order to generate a current, unique view for a user. Or a portal application channel could simply put a user-facing wrapper on a Web service. In short, portals are, or can be consumers of a veritable alphabet soup of XML specifications.

Internally, portals sometimes incorporate technologies that allow them to produce output that is independent of any particular browser or device, such as the Universal Interface Markup Language (UIML) or Mozilla's XML User Interface Language (XUL). Many portals also produce browser or device-dependent content using various XML specifications for presenting information to browsers and portable computing devices such as XHTML or Wireless Markup Language (WML) output. Platform independent user interface description languages are still in their infancy. Most portals tend to rely on a presentation technology such as XSL to generate an appropriate view of the portal based upon information about the software and device making the request.

Portals also have the potential to play a role in direct competition with Microsoft's attempt to position Windows as a platform for digital rights management. A portal can interpret and apply rules coded in digital rights management XML specifications (e.g. XrML – extensible rights Markup Language) that outline who should have access to what data. By incorporating browser plug-ins, a portal can also decrypt and display encrypted and/or multimedia content. Indeed, by providing just-in-time access to the viewer applications, portals provide a platform-agnostic alternative to proprietary OS-bound digital rights management solutions. But it will be up to content providers and open source developers to ensure that portals are able to fulfill their potential in the realm of digital rights management.

XML and Identity, Authorization and Access Control

A portal works its magic by knowing who you are and discovering your role in and relationship to an organization. It knows who you are when you log in. At the lowest end of the authentication spectrum, it may simply retrieve and view the contents of a cookie that has been stored by your browser on your desktop workstation from a past session, or may ask you to supply a userid and password which it checks directly or presents to a directory for confirmation that the credentials match what it has stored for you. Since portals often provide single sign-on access, are integrated with other enterprise information and communication systems, and have access to personalized data and services, a portal may require stronger authentication credentials for some or all

transactions. These might take the form of a digital certificate stored on a smart card, plus a personal identification number (something you have and something you know), or biometric data such as retinal scan or fingerprint scan (something you are).

A portal uses data about users to make decisions about content and access, and this data is often supplied by directories implemented using the Lightweight Directory Access Protocol (LDAP). LDAP is a specialized database designed for storage and rapid retrieval of data about people and resources. A portal can function as an LDAP client and make LDAP queries directly, but it is becoming increasingly common for applications to exchange data with directories using XML. The Directory Services Markup Language (DSML) is a common XML specification that defines both a way to describe directory data and a protocol for interacting with directories.

Example 1-1
A DSML user record provided by a middleware layer that communicates with an LDAP directory

Portal integrated applications need access to data about a user to make personalization or authorization decisions and this is something that a portal framework can provide to them. The ability of a portal to provide secure, seamless integration with other applications is attractive to users and developers. It eliminates the need for every application to support authentication, which in turn means that once the user has logged into a portal they do not have to log into integrated applications. There are many approaches to single sign-on. The Sun Microsystems-sponsored Liberty Alliance uses the Security Assertion Markup Language (SAML) to share identity information and manage authorization between services through a federated network identity. Liberty is a single sign-on specification for sharing information and a platform independent way to exchange it. The Liberty Alliance relies upon open specifications and has significant and growing support in the vendor and open source communities.

Federated identity implies that no one system owns the user's identity, but this has benefits and drawbacks. The model proposed by the Liberty Alliance incorporates the concept of an identity provider that handles user authentication. When a portal plays the role of identity provider it offers an anchor for the circle of trust environment. Users who pass through the entrance have the convenience of single sign-on. The portal user interface persists and serves to remind users that they are in a trusted environment. The portal can also provide developer hooks to make it easier to integrate applications with the single sign-on service and provide services like global session management and access to core user data if available and permitted. So single sign-on bound with a portal can be a good fit for both users and developers. Users can more easily grasp the concept that while they are "in" the portal, they have access to many applications, whereas unbounded single sign-on such as the Microsoft Passport model can pose a significant security risk if users fail to understand the concept and leave themselves logged into the framework after exiting an integrated application such as Hotmail. The portal can manage a global session for users, and end sessions that the user has established with integrated applications when they log out of the portal. Developers can build channels and applications without having to build an authentication mechanism. They can also use a

consistent API for specifying application interactions with the single sign-on and portal frameworks.

Rules that govern access to services and information resources such as elements of a user model may also be encoded in XML. The eXtensible Access Control Markup Language (XACML) is a specification for expressing information access policies for Internet accessible resources and data. Each policy statement describes a subject, an object, and an action. A subject might indicate a userid, group id or a role. Objects can be an XML document or a specific element within an XML document. Basic actions include read, write, create, and delete. This allows for very fine-grained access control management for XML content. The XACML standard also specifies a request-response protocol for retrieving XACML policy data. XACML can be used to manage policies related to Web services. For example, XACML policy statements could be used to describe policies for submitting requests and receiving responses from individual Web services. There is strong vendor support for XACML with both IBM and Sun providing toolkits to facilitate development of XACML policies and applications.

XML for Content Encoding, Metadata, Classification, Rights management

Portals may be among the first applications to realize the dream of the semantic Web. When a channel is defined and registered with a portal framework, its content can be described using machine-readable metadata elements specified with RDF (Resource Description Framework). This structured metadata can be useful to end-users when they are trying to locate a channel that performs a specific task, and it can also be used by software to programmatically evaluate content. The framework can perform any number of tasks such as identifying new content and recommending it to end users, if it fits a set of requirements that define the end user's interests. This data can also be used by the portal framework's personalization logic to build or expand upon page templates for users. Metadata that captures semantic information about an object in a machine-readable format can enable many proactive end user information services.

There are numerous special purpose metadata specifications defined in XML. In a vertical portal, domain-specific metadata can be very beneficial. For example, a medical portal might contain channels with medicine oriented metadata while a higher education portal might include content described using metadata for describing online courses and allow users to browse and search using elements from these specifications. The World Wide Web consortium provides a general-purpose metadata specification called Dublin Core, which is appropriate for describing a wide range of electronic resources. It contains 15 basic elements such as title, keywords, creator, and description. Most of these elements can contain any appropriate free-form content defined by the author or classifier of the content. Example 1-2 provides an example of Dublin Core metadata embedded in a channel.

The Dublin Core metadata element set includes an element called rights, intended to support content access and usage metadata. Like most other Dublin Core elements, there

is no well-defined, widely accepted collection of values allowed within the element. This is both a strength and a weakness of the Dublin Core. In the case of the rights element, which is intended to specify information such as who can access the content, the lack of agreed-upon values for this field hinders its usability across multiple organizations and applications.

An XML digital rights metadata specification called the Extensible Rights Markup Language (XrML) was developed to address the need for specifying detailed rights metadata. Since a portal has access to content and metadata and is in a position to know something about a user, it can assume a role as arbitrator of content access requests, and programmatically ensure that XrML rules governing content access and use are obeyed. Because it accommodates digital signatures and encryption keys that control access to content, XrML is especially well suited to managing digital rights as they relate to commercial, multimedia content such as audio or video, although it is also used to manage access to electronic text and Web Services. The specification is publicly available, although some uses require that the developer pay the originator of the specification, ContentGuard, a licensing fee. For this reason, developers may prefer to utilize another specification such as XACML (described in the next section). But since many vendors and some standards now explicitly support XrML, it is an important content metadata specification that a content aggregation platform such as a portal will likely need to support.

There are XML specifications that can be used to record a control vocabulary and the relationship between terms, also referred to as a taxonomy or ontology. An ontology contains terms that belong to a particular knowledge domain – that is they are accepted words or phrases for describing concepts. They are often represented as hierarchies, although there is sometimes a fair amount of cross-linking between categories. XFML (XML Facet Markup Language) represents discrete concepts as facets and provides a framework for describing and sharing facet collections. The Web Ontology Language (OWL) is another specification concerned with creation of machine readable ontologies to support the Semantic Web. Ontologies can be used by metadata authors to ensure that keywords and other metadata content are consistently represented across multiple resources. This consistency makes it easier for users to find items, because they can browse or search on these terms, and it allows for software to programmatically evaluate the semantics of a resource. So metadata and ontologies are complimentary concepts that depend on one another to provide maximum value to end-users.

Most channel content is described using structured XML data. Depending on the portal framework, any number of content formats might be supported by a portal. Indeed, the trend in portal frameworks is to support any XML specification as long as the developer can supply a transformation that allows it to be rendered for Web browsers, cell phones, and other devices. An easy way to migrate existing content into a portal framework is to simply upgrade the content from HTML to XHTML. Many portal frameworks provide various mechanisms for rendering XHTML content.

RSS (Rich or RDF Site Summary), a content format specifically designed for describing syndicated content, is another alternative. RSS is best suited for information channels, as it is limited to presenting link-description pairs. News headlines, bookmarks and frequently used links, and tables of contents can all be easily tagged using RSS. RSS is a concise descriptive format that is well suited for encapsulating informational channel content for portals. Syndication implies that data is updated periodically but RSS does not include elements for specifying this information. But it does support a set of RDF-specified metadata elements called the Dublin Core syndication module that do specify syndication information such as the frequency with which content is updated and when it expires. RSS and its modules are described in more detail in chapter ??.

Example 1-2
An RSS 1.0 code example, with Dublin Core encoded metadata embedded using RDF

As a content consumer, a portal needs to interface with or provide an integrated content management solution. Content management systems serve as repositories for various types of electronic documents, control who has access to content and when (workflow), and provide tools for content editing. Like other applications, a CMS can be integrated via single sign-on so that the portal handles authorization and authentication. Portals can use group membership to manage content access as well, and provide channels that proactively facilitate content management such as expiration notifications and workflow event notification. For example, a portal channel might remind a particular user about new content that is available for review or awaiting approval. For a portal, a content management solution with a rich set of tools for storing, editing, managing and transforming XML content is essential. Such a CMS can serve as a repository for local content, metadata about local, remote and dynamic content, taxonomies, syndication metadata, as well as rights metadata, Web services interfaces, and access policies.

XML Specifications for Application Integration

Portal channels can serve as enterprise to end user integration solutions when combined with integration and interface XML specifications that are platform and language independent. Integration can and does happen at the framework level as we've already seen, but the integration that occurs within channels is what allows services to be defined and published to end users. Here again middleware plays a prominent role. Middleware in the form of Web services represent services that can be exposed directly or indirectly through portal channels. Channels can also be consumers or producers of messages to other systems. Truly platform independent channels and Web services provide output or user interface elements in platform independent formats and rely on the portal framework to use technologies such as XSL and XSLT to ensure that the results are formatted appropriately for the browser and device.

Portal data, information, and application channels can all be consumers of Web services. Channels could be implemented so that they discover Web Services through a directory using the Universal Description and Discovery Interface (UDDI) and retrieve Web Services Description Language (WSDL) documents that describe the actual interface to the service. WSDL specifies both the arguments and function calls available in the Web

service and the responses. Or if a channel is communicating with an RPC-style Web service then it will already have all the information it needs in order to communicate with the service. The channel can then communicate with the Web service via the XML-based Simple Object Access Protocol (SOAP) or XML-RPC. In this way portals can aggregate not only local but also remote services available anywhere on the Internet.

A channel might also support Web service chaining, where output of one Web service is supplied to another as input until a task is completed. Automatic travel scheduling is one often-used example of Web services chaining to perform a series of tasks where data from one service is needed by another. The scenario starts with one service consulting a user's personal calendar and then supplying potential dates to a travel agency Web Service in order to schedule transportation, lodging, or car rentals.

Messaging is another option for integrating applications with a portal. Messaging involves one application sending some information to another, sometimes with the assistance of an intermediary to route messages and ensure delivery. A message also requires a destination address, so it knows where to go. Web services is a messaging solution, albeit an extremely simple one. SOAP acts as the message container, HTTP as the transport mechanism, and a URL as the destination address. Messaging can basically employ any transport and any parseable message format. With these requirements, even e-mail and chat can be used for inter-application messaging. And in fact, there is an open protocol that supports both traditional chat facilities and inter-application messaging, called Extensible Messaging and Presence Protocol (XMPP). An open source chat utility called Jabber is based on XMPP. Since both application messaging and chat facilities are both highly desirable capabilities for a portal, XMPP is another XML specification that portals will need to use and understand. While there are many other ways to implement messaging, this book will focus on SOAP and XMPP, since they are open, multi-purpose standards.

One challenging obstacle to enterprise to end user integration is describing the user interface in a browser and platform-independent way. It is unfortunately still all too common to find HTML elements hard coded into Web applications and even some portals. When presentation issues are handled by the same code that manages logic or data for an application, it becomes impossible to reuse the code for any other purpose. Increasingly portal users are using not just Web browsers, but also PDAs and cell phones which may have their own requirements for formatting and presentation capabilities that differ from Web browsers on desktop workstations. Such poorly designed applications must be rewritten in order to function within a portal environment.

Converting legacy applications or applications where presentation and logic data are combined are more challenging depending on whether the presentation logic is formatting output or actually contains interactive user interface elements (e.g. HTML form elements). In cases where the interface is merely the static output of some task, then it becomes easier to convert the application. Here the post-query logic is primarily concerned with rendering so the code that wraps the output just needs to be updated to use the appropriate browser-independent elements.

In the case of program code that supports ongoing interaction between the user and the application, the ideal solution is to describe the user interface elements or widgets in a platform and browser-independent way. There are a couple of ways to do this. One is to describe the interface in some browser-independent language and then use a code generation tool to create functions or methods that produce platform specific user interface elements. The User Interface Markup Language (UIML) is especially well suited for this task. Its elements describe interface components such as form fields and buttons as discrete parts, which can be assembled in various ways and can contain content (labels, default values, etc.). It strives to describe elements without focusing on any specific user interface toolkits. Another option is to generate a browser-neutral interface at runtime and let the portal framework transform this data into the appropriate format for the browser. The open source Mozilla project uses the XML User-interface Language (XUL) to describe the Mozilla Web browser user interface, so it is designed for providing platform-independent interface descriptions that can be transformed as needed in real-time. XUL supports a rich collection of user interface elements and UI events that a user can trigger while interacting with an application.

Directory and middleware XML specifications including DSML, SAML, and XACML enable personalization, authorization, and single sign-on. Structured content and metadata specifications provide users with dynamic, interactive, self-refreshing channels and options for finding and using channels. Web services and Web Services for Remote Portals provide the backend integration that makes it possible to build dynamic content that can be delivered through portals or is specifically designed for portals. GUI description specifications like XUL and UIML allow for true presentation-logic separation. XSL and XSLT handle channel and framework transformation allowing portals to offer services to any device and browser. XML is truly essential to portals.