Introduction to Portals and Portlets

Green Paper from

Portlets in Action
EARLY ACCESS EDITION
AUTHOR(S) Ashish Sarin

MEAP Release: October 2009
Softbound print: Fall 2010 (est.) | 475 pages
ISBN: 9781935182542

This green paper is taken from the book Portlets in Action from Manning Publications. It introduces the essential concepts and benefits of Java Portlet technology. For the table of contents, the Author Forum, and other resources, go to http://manning.com/sarin/.

When the internet first came about, ‘content’ reigned supreme. And then, ‘user experience’ took over. If you have been an internet user for last couple of years, this transformation wouldn’t go unnoticed. In this transformation phase, intended to attract users by enriching user experience, most websites got a face-lift in the form of user customizable themes, and features to allow users to control how the content is presented and what content is presented.

The lack of a standard approach and technology to address the user experience requirements like personalization, customization and content aggregation in a web application led to ad hoc ways to implement such features, resulting in maintenance nightmares, lost developer productivity and longer turnaround time to incorporate new features for enriching user experience. With the arrival of Java Portlet technology, the landscape of user experience aspect in internet applications is set to change. Java Portlet technology provides a standard approach to incorporating user experience features in your web application, which includes consistent look and feel, personalization, customization and content aggregation.

What is Portal?

If you have been using internet then you already know something about portals. It’s not a new concept but a more formal definition didn’t come into existence until recently.

A portal is a collection of mini web applications, called portlets, which supports features like personalization, content aggregation, authentication and customization. Portlets act as windowed web applications within the portal and each window on a portal web page (called portal page) represents a portlet.

To get a feel of portals, you can visit iGoogle (http://www.google.com/ig) portal to discover the features offered by a portal. Figure 1 shows home page of the iGoogle portal after a user logs into the portal. The home page of iGoogle portal has a portlet for showing headlines from CNN, a portlet for showing / searching information from YouTube, and so on. The portlets can be personalized by users, for example, users can change the number of emails they want to view in the portlet, number of CNN headlines news they want to view or set the location from

1 ©Manning Publications Co. For Source Code, Sample Chapters, the Author Forum and other resources go to http://www.manning.com/Sarin
The users can drag and drop these portlet windows on the portal page to customize the way information is organized on the page. The user can also choose to add more portlets or remove one or more of them from a portal page.

The take-away point from figure 1 is the user gets the unified view of information from different information sources. This is similar to a TV showing different channels in distinct windows on the screen. Portlets have been referred to as ‘mini web applications’ here because portlets provide limited information and features to the user compared to the original web application they represent. For example, in figure 1, Time Card portlet displays information about due and missing time card entries, but it doesn’t show information about approved time cards or allows users to submit a new time card, which is provided by the original Time Card application.

The core business functionality provided by a portal is no different from what a set of distinct web applications would provide. In figure 1, the content (information or service) provided by Gmail, YouTube and CNN portlet is also provided by Gmail, CNN and YouTube applications. If the business functionality remains the same then what’s the business value in creating a portal? In figure 1 the information displayed to the user comes from disparate data sources (as depicted in figure 2), and the portal aggregates the information to provide a consolidated view to the user.

---

2 ©Manning Publications Co. For Source Code, Sample Chapters, the Author Forum and other resources go to http://www.manning.com/Sarin
The use of portal to aggregate content from different sources results in increased efficiency and enriched user experience as the user doesn’t need to go to distinct web applications to access content. In figure 2, the portlets on the home page of iGoogle provide most frequently accessed information or feature of the applications they represent.

Now that we know about portals, let’s see the benefits of using portals compared to a stack of distinct web applications.

**Portal benefits**

Developing a web portal makes a good business case if it’s required to gather and present information from various data sources, applications and systems, to give a unified view of the information to the user based on his identity. And, it is expected that the user experience is enhanced by a single point of entry to the information and flexibility to the user to customize and personalize the information.

**Enriched User Experience**

Let’s say, as an employee of an organization you need to frequently access organization-specific B2E applications (like Time Card, Help Desk, Knowledge Management and Service Request) to know about your missing time cards, recently published articles, closed help desk tickets, and so on. These applications have their own data sources from which the information is obtained or stored by each of these web applications. You will usually go to each of these different applications to know about missing /due time cards, recently published articles and status of previously submitted help desk tickets. This interaction between employee and B2E applications is represented in Figure 3.

This isn’t an optimized way of accessing information and services because you need to go to different web applications and authenticate. An intranet site which provides single sign-on feature and access to all these different applications would have been a better solution.

---

3 ©Manning Publications Co. For Source Code, Sample Chapters, the Author Forum and other resources go to http://www.manning.com/Sarin
Let’s say, the organization goes one step ahead and provides a single sign-on solution and access to the different web applications from an intranet website, as represented in Figure 4. By providing single sign-on feature, the organization has provided easy access to the B2E applications, but you still need to filter the information that interests you. For example, if you are interested only in automobile related articles then you will have search for the articles in the Knowledge Management application, if you are only interested in your open help desk tickets then you need to search for it in the HelpDesk web application, and so on. These individual B2E applications may provide some level of personalization and customization based on your identity and preferences.

The personalization and customization features, embedded in individual B2E applications, and single entry point to these applications from an intranet website enhances the user experience but it still fails to provide a unified view of information and easy access to services offered by different B2E applications. For instance, you still need to go to Help Desk application to access its content (which includes information and services offered by Help Desk application).

The ideal scenario for an employee is to view information from distinct B2E applications in a single application and should be able to access the most commonly used services of B2E applications from a single application. This scenario is represented in figure 5 and is achieved using intranet portals. In figure 5, the user mostly interacts with the portal and in some cases with the original web application also. For instance, the portal directs the user to the Knowledge Management web application if certain specific features or information are requested by the user.
Figure 5: Organization’s intranet Portal gathers content from different data sources, which means that users don’t need to access different web applications for information or services. In figure 1 the intranet website simply redirected the users to the original web application but when using portals, the user is taken to the original web application only if some least-accessed information or services are requested by the user.

Usually, portals provide the most-used features of the original web application to the user and when least-used features of the web application are requested then the portal redirects the user to the original web application for performing such action(s). It is possible that the portal doesn’t even show the least-used features of the web application to the user; instead give an option to the user to visit the original web application.

**NOTE**

The design of the existing systems plays an important role in how quickly business can start using portal. In Figure 5, we see that information for knowledge management and time card application comes from the databases that were used by these applications, which requires re-coding the business logic. On the other hand, HelpDesk application business logic is accessed using Web Services as the application was designed as a set of services which can be reused by other applications – an approach defined by Service Oriented Architecture (SOA). It is recommended that applications should internally make use of SOA as it allows exposing business functionality as a service in future, saving the effort to redo the business logic.

Portlets in a web portal provide limited content compared to dedicated web applications they represent, which means portals have to be used along with web applications to provide content to the user.

**Compliments Web Applications**

Portals are not a replacement for web applications, but are meant to extend the functionality of existing web applications by gathering relevant content from the existing information systems and displaying it to the user based on their identity and preferences. When certain information or features are requested by the user, the portal redirects the users to the original web applications.

**BRIDGE INFORMATION GAP**

A web application mainly targets a part of the business process and not the complete business, for example, there will be a web application for each of the following business processes: customer management, inventory, order processing, and so on. These web applications give disjointed view of the business as the inventory application doesn’t know about orders placed and customer management application doesn’t know about inventory status. These web applications don’t need to know about information managed by other web applications but a business user may want to know the information from all these different applications to get a unified view of the business.

A web portal bridges the information gap that exists between these individual web applications by bringing information from the different data sources used by these web applications, and showing it to a business user, thus giving a unified view of the business.

**INTER-PORTLET COMMUNICATION**

Web applications communicate with each other using database or messaging middlewares. The result of such communication is not visible immediately and requires users to access the web applications separately to view the affect of the communicated information.

In most cases, the portlets in a portal may interact with each other in response to user actions, to show the relevant information based on user actions. For instance, in a ‘Weather Portal’, if a user selects the city from the
Location portlet, the Weather portlet updates the content to display the next 5-day weather forecast for the selected city.

**One size doesn’t fit all**

Portals are not answer to every business requirements and organizations should carefully choose whether there is a business case for developing a portal. If the business requirement doesn’t require gathering content from distinct information systems, that is, effectively integrating disparate systems loosely, then businesses should consider developing an independent web application to meet the requirements.

The personalization and customization features in portals are important from the user’s perspective and from business perspective, the most important requirement to consider for the portal is content aggregation.

A portal consists of multiple portal pages, similar to web applications which consist of multiple web pages. The only way in which content can be added to a portal page is via portlets.

**What is Portlet?**

A portlet is a pluggable user interface component which provides specific piece of content, which could be a service or information from existing information systems. Portlet components are responsible for providing the user interface of the portal by accessing distinct applications, systems or data sources and generating markup fragment to present the content to the portal users. Some of the examples of portlets are: a weather portlet which provides weather information of a city by accessing Yahoo! Weather RSS Feed service and a HelpDesk portlet which shows the pending helpdesk tickets information from helpdesk database.

The primary responsibility of a portlet is to generate markup fragment (like HTML, XML and WML), which is then displayed on a portal page within a window, called a portlet window. A portal page usually contains multiple portlets and displays them in distinct portlet windows, each having its own title and a set of buttons to change its look and feel, preferences, maximize and minimize window. The figure 6 shows a magnified view of what a portlet window looks like when rendered on a portal page. The portlet options, like changing window state, preferences and configuration, available for a portlet may vary based on the user permissions in the portal and the configuration of the portlet in the portal.

![Figure 6: Article portlet showing buttons to configure, minimize, maximize and close window. The options shown for a portlet vary depending upon its configuration and access permissions to the user.](image)

The definition of portlet components mentions that they are ‘pluggable user interface’ components, which means a portlet component is responsible for generating the user interface specific to that component, unlike Servlets components where user interface is not componentized. So what makes portlets ‘pluggable user interface’ components?
**Pluggable User Interface**

A portlet generates markup fragment, and not the complete portal page. The responsibility of displaying the complete portal page is with the portal server which aggregates fragments generated by portlets and displaying them in distinct windows on the portal page. This division of responsibility makes it possible for a portlet component to focus on generating the user interface that is specific to the component. Like any software component, portlet components can be combined with other portlet components to create a web portal, bringing pluggability to the user interface. For instance, a NEWS portlet which shows news from CNN website can be used in any web portal which intends to show news from CNN.

**Portlet 1.0 Vs Portlet 2.0 specification**

The two specifications of portlets that currently exists are: 1.0 and 2.0. The 2.0 specification addresses most of the frequently required features of portlets that were missing from 1.0 specification, like resource serving, inter-portlet communication and portlet filters. Most portlet containers support both the specifications and the 2.0 specification is backward compatible with 1.0 specification.

Portlet components generate user interface and so do the servlet components. In the next section we discuss the reasons for choosing portlet components over servlets.

**Why Portlets?**

In Service Oriented Architecture (SOA), service orchestration (or collaboration) makes it possible to develop applications from existing services. Portlets play a crucial role in developing SOA based applications by providing interaction between portlets at the user interface layer.

A portlet component in a web portal generates the user interface for the information or service it encapsulates. If the services represented by portlets in a portal need to interact with each other for service orchestration then it’s required for portlets to communicate. The portlet specification provides means to achieve this inter-portlet communication using events and public render parameters. The portlet container is responsible for handling the communication between portlets, making communicating portlets independent of each other and so are the distinct services they represent. For example, let’s say you have a LOCATION portlet, a WEATHER portlet and a BUSINESSES portlet where:

- **LOCATION** portlet represents a service which allows users to search and select a city regarding which they would like to know weather information and the businesses in that city.
- **WEATHER** portlet provides weather information service based on city
- **BUSINESSES** portlet provides the information about the businesses in the city

The location, weather and businesses services are distinct services and each of the portlet wrapping these services generate user interface specific to the service, as shown in figure 7. For instance, LOCATION portlet will generate user interface which allows users to search and select a city and BUSINESSES portlet will display information about the business based on pre-defined categories.
The LOCATION portlet can interact with WEATHER and BUSINESSES portlets using inter-portlet communication mechanisms provided by the portlet container, resulting in seamless integration between services provided by the portlets.

Portlets generate the complete web page and not a markup fragment, and they cannot be used to work in collaboration with other servlets to generate the complete web page. Servlets can be designed to encapsulate a service but it will be a standalone service because of the lack of support from the web container to allow communication between servlets without being directly dependent upon each other.

Now that we know what a portal is and how portlets make up the content for a portal, let’s really see what we need to deploy portals.

**Portal Infrastructure**

When creating web applications using Servlets, the web container provides the environment which manages the Servlet components and web server is responsible for serving web pages to the web browser. Similarly, portlet components are managed by portlet container and portal server is responsible for serving portal pages to the web browser.

Irrespective of the portal server you are using, the basic steps that are required for creating a portlet and to see it in action include:

- Installing the portal server and portlet container. In most cases portlet container and portal server components are packaged together as a single component.
- Writing portlet class which contains the logic to generate the markup fragment
- Creating portlet configuration files to register portlet with the portal server
- Creating portlet deployment descriptor
- Packaging portlets in a WAR file using a build tool like ANT or Maven.
- Deploying the portlet WAR file on the portal server
- Creating portal pages, which you can think of as blank web pages with no content
- Adding portlets to the portal page, which provide content for the portal page

**Portlet Container**

A portlet on a portal page is represented by a portlet instance inside the portlet container. Figure 8 shows the portal infrastructure components and how they fit together.
Figure 8: Portal infrastructure consists of portal server and portlet container. The portlet container manages the portlet instances and hands over the markup fragments generated by the portlets to the portal server. The portal server aggregates the markup fragments to create the portal page.

Portlet container’s responsibilities include managing portlet instances and handing over the fragments generated by the portlets to the portal server for aggregation.

**LIFECYCLE MANAGEMENT**

The portlet container is responsible for invoking lifecycle methods on the portlet instances and providing them with the required runtime environment. A portlet container is an extension to servlet container, that is, it provides what a servlet container provides, and additionally manages the portlet instances. The portlets access their runtime environment using `PortletContext` object (similar to `ServletContext`), which allows portlets to share data with other portlets and Servlets in the same portlet application.

**PORTABILITY**

A portlet container must follow the requirements laid down by the Portlet 2.0 specification, for portlets to be portable across portlet containers. Most portlet containers provide extension to the base set of requirements detailed in the specification and it is the decision of the portlet developer to use such features as it will make portlets non-compliant with the specification and hence not portable across different portlet containers. If you are developing portlets targeted to a specific portlet container and it’s not required for portlets to be portable across different portlet containers, then use of container-specific extensions should be considered.

**Portal Server**

A portal server is responsible for submitting user requests received from the portal page to the portlet container and aggregating response generated by portlets to form the portal page. A portal server is responsible for generating the portal page; therefore, the responsibility of providing consistent look and feel for the portal lies with the portal server.

**NOTE**

You can consider a portal server as a component that sits between the user request from the portal page and the portlet container.

A portlet container is not responsible for generating the portal page; it hands over the content generated by the portlets to the portal server for aggregating and displaying the portal page. Figure 9 shows interaction between portal page, portal server, portlet container and portlet instance when handling a portlet request.
There are many open source (like Liferay and JBoss Portal) and commercial (like IBM WebSphere Portal) portal servers which provide a wide range of features (like administration, content management, search, single sign-on and so on) to the portal developers, making it easy to quickly setup a fully-functional portal. The choice of a portal server plays an important role in portal projects, as it can help you reduce the development time by providing built-in portlets, integration with external systems (like LDAP, SAP, and so on) and flexibility to access portal server functionality from your custom portlets.

**CHOOSING A PORTAL SERVER**

The biggest assets of a portal server are the built-in portlets that it provides. The choice of a portal server should be driven by comparing the portal requirements and comparing them with the features that are provided out-of-the-box by the portal server. For instance, if your portal requires a discussion forum, a document library and announcement functionality then evaluate portal servers based on the availability of built-in portlets for these functionalities.

The requirements of a portal may change over a period of time as the business requirements evolve, making it difficult to implement certain functionalities that were not initially evaluated when selecting the portal server. To take care of such scenarios, portal severs should also be evaluated in terms of flexibility they provide in customizing built-in portlet’s behavior and information it displays and access to portal server functionality that can be used by portlets to address new business requirements.

**Summary**

The portlet technology is promising in terms of its reach and impact. The focus on portals has increased in the recent years and it has gained momentum with the release of Portlet 2.0 specification. It is important to understand the unique features (personalization, content aggregation, customization and authentication) of a portal to get a feel of how Java Portlet technology makes it easy to develop portals.