Oracle® Fusion Middleware

Overview for Oracle WebLogic Portal

10*g* Release 3 (10.3.4)

E14241-02

November 2011

This guide includes these sections:

1 About the Oracle WebLogic Portal Documentation

Welcome to Oracle WebLogic Portal. The WebLogic Portal documentation provides you with in-depth information about specific feature areas and guides you through the portal development and management life cycle. Full feature-level guides are available on Oracle Technology Network (OTN), listed in the Features View column. Those same guides are also presented in the Life Cycle View path according to each phase of the portal life cycle.

Developing portals with WebLogic Portal is an iterative process characterized by four phases: architecture, development, staging (assemble/deploy), and production (management of your portal).

Each phase in the portal life cycle involves specific tasks and tools. The WebLogic Portal documentation maps to the portal life cycle and provides a context-rich environment for learning about building portal applications.

By reviewing the WebLogic Portal documentation according to the portal life cycle, you can understand the big-picture context, and at the same time learn about the deeper, more detailed areas you need to dive into, so that you always know where you are and how your efforts relate to other phases. To accomplish this contextual mapping, the documentation provides connection points to other phases and other related feature areas, letting you enter the documentation anywhere.

For example, in the development phase, you create your application files and functionality using the Oracle Enterprise Pack for Eclipse (OEPE) IDE, your source control system, and other tools of choice for creating portal application resources (Java source code, JSPs, XML, JavaScript, graphics, and so on). In the staging phase, you use the WebLogic Portal Administration Console to create and define access to your portals, fine tune portal functionality, create content, and move your application from one environment to another (such as from staging to production).

That said, Oracle recommends a top-down approach to begin: by first reviewing this Overview, especially if you are new to WebLogic Portal.

2 Introduction to WebLogic Portal

WebLogic Portal is the outward-facing component of Oracle WebLogic Platform that lets you provide a user interface to integrate your applications. WebLogic Portal lets you surface application data and functionality from heterogeneous environments into an integrated, dynamic, and customizable Web-based portal user interface that can simultaneously support your customers, partners, and employees on multiple devices. WebLogic Portal handles the infrastructure so that you can focus your development efforts on what is most important — your applications.

In addition to a powerful portal framework and its J2EE security foundation, WebLogic Portal provides many business services, such as content management, communities, personalization, search, and user management.

WebLogic Portal provides a virtual content repository that lets you federate external content management systems into a single management interface. You can then build portals using content in those external resource. WebLogic Portal also provides a WLP content repository for creating and managing content.

This guide is the starting point for understanding and developing applications with WebLogic Portal. This chapter introduces portal concepts, describes the WebLogic Portal infrastructure framework and business services, and describes the portal life cycle, from architecture through development, staging, and production. The remaining chapters in this guide describe each phase of the life cycle in detail and provide a starting point for using WebLogic Portal.

This chapter includes the following sections:

2.1 Portal Concepts

Portals let you combine discrete processes and pieces of functionality into a single Web interface. Before portals, Web users could visit only one page at a time. They could, for example, search for airline flights in a browser window, but if they wanted to see their personal calendars or view their organization's travel policies, they would have to open a new window and probably log in more than once. Chances are that each site would also look and feel differently than the other sites. With portals, users are able to view all of those sites, and many more, in a single browser window with single sign-on and a common look and feel.

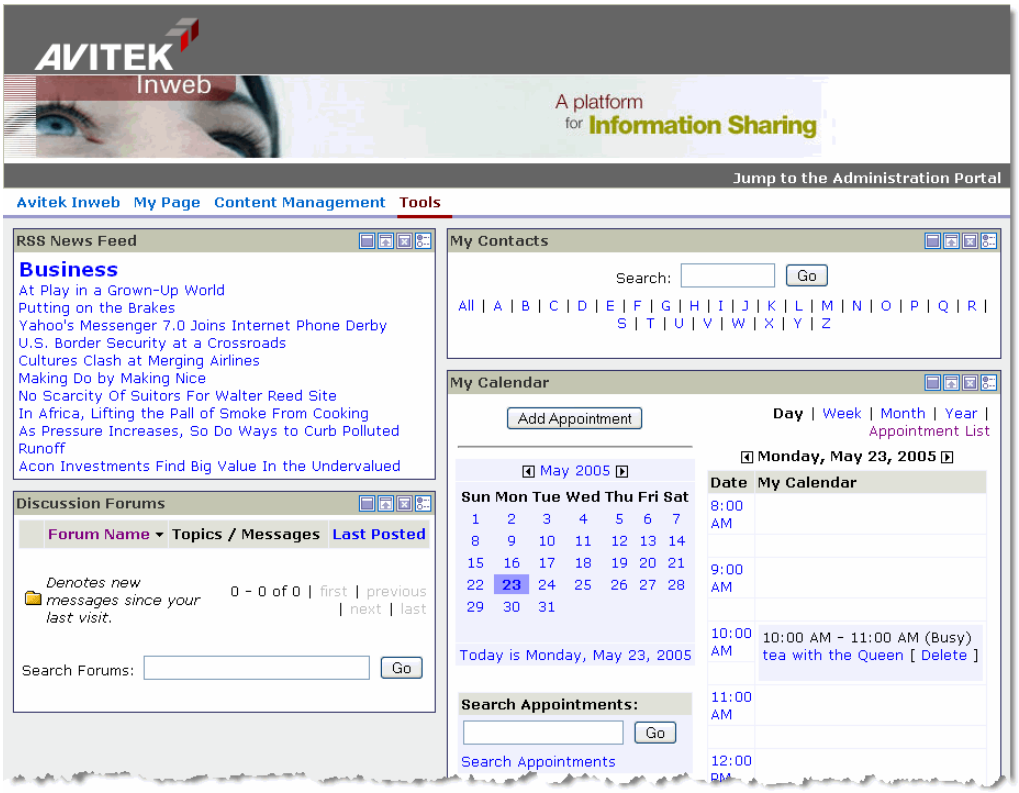
Portals accomplish this unified view with individual windows called portlets. Each portlet can contain a process flow, functionality, or content, and multiple portlets can be displayed in a single browser window. In some portals, portlets can even communicate with each other. Portlets are grouped onto a page. A page is a viewable area represented by a clickable link often in the form of a tab. Pages can further be grouped into a book for organization. Books are also represented by navigation links or tabs. In all, a portal can contain multiple books, each with multiple pages that display multiple portlets on each page.

Another characteristic of a portal is a common look and feel shared among books, pages, and portlets. In some cases users can "re-skin" a portal by changing the look and feel from a drop-down field or with some other control.

A portal often uses a single security framework for authentication and authorization, even when a portal accesses external data (such as RSS news feeds), external business processes (Web Services), and external portlets (Web Services for Remote Portlets [WSRP]).

Figure 1 shows a page that contains a number of portlets and other functionality. For example, the desktop header could display a campaign, one portlet could encompass a complex process flow and communicate with another portlet, and another portlet could come from a remote portal.

Figure 1 A Portal Desktop



Technically speaking, portals are collections of resources in an enterprise application that can be displayed in customizable, personalized, and audience-specific views called desktops. For example, a portal might contain business logic, process flows, and content for internal employees as well as for partners. Portal administrators could create an employee desktop and a partner desktop that surfaces the appropriate flows and content for each audience. With a unified security framework, such as that provided by Java 2 Enterprise Edition (J2EE) and WebLogic Server, along with a flexible entitlements engine (WebLogic Portal), a single desktop can also contain all resources and dynamically display only the resources for which an authenticated user is entitled. For example, a desktop could contain pages and portlets for both employees and partners, but a partner who logs in does not see the employee pages and portlets, and vice versa.

Reuse is also a key portal concept. For example, a single process flow in a portal can be reused in multiple portlets in multiple desktops.

2.2 WebLogic Portal Framework

A portal desktop, when viewed in a browser, is simply HTML code (along with browser-supported resources such as graphics, animations, cascading style sheets, and JavaScript). The key to the dynamic nature and functional power of a portal is the behind-the-scenes processing that occurs at each click or selection.

Traditional behind-the-scenes processing (in the Java world) has included client-side functionality such as applets, and server-side functionality such as JSPs, servlets, EJBs, and other J2EE features. All of these features perform their processing whether or not a portal is involved. WebLogic Portal provides a framework that dynamically constructs your portal user interface on each request using standards-based components (such as XML, JSPs, JavaScript, cascading style sheets, and GIFs).

The WebLogic Portal framework lets you easily customize and reuse your portal user interfaces for multiple types of devices simultaneously. With your portal user interface in place, you can surface your application functionality in an integrated view. For example, you can create a portal look and feel that displays your portal application a specific way in a desktop browser, and you can create a sub-look and feel that displays the portal and its content automatically in a completely different way when accessed by a handheld device.

The WebLogic Portal framework uses the WSRP standard to handle federated portals, letting you produce books, pages, and portlets that can be consumed by remote portals, and consume books, pages, and portlets that are hosted by remote portals.

WebLogic Portal also provides a framework that lets users create their own community portals, where they can assemble the portlets they need to work together interactively in groups. Communities let users create their own secure portal desktops that group members use to communicate with each other, create and share content, invite new users to join, and manage their own community.

In addition to enhancing collaboration with communities, WebLogic Portal also lets you rebrand the tools in the WebLogic Portal Administration Console for use in your own custom applications. WebLogic Portal lets you surface pieces of management functionality and combine them with other application functionality to create many types of applications, including executive dashboards, collaboration communities, ASP affiliate applications, ISV-branded applications, self-service applications, and OA&M applications.

The services provided by the WebLogic Portal framework let you develop and manage your applications easily and more rapidly, increasing productivity and return on investment while enhancing the user experience.

2.3 WebLogic Portal Business Services

In addition to a powerful framework for portal user interface development and application integration, WebLogic Portal provides the following services that can make your applications more powerful and personalized.

- **Access Control** - You can provide dynamic access control to portal components and administration tools.

  With visitor entitlements, you can determine which users can access specific desktops, books, pages, portlets, and look and feels. For example, in an HR

desktop, you can entitle only managers to see manager-specific portlets; or you can entitle a desktop so that only partners can access it.

With delegated administration, you can determine which administrative users can access specific WebLogic Portal Administration Console tools and set different levels of administrative rights on those tools. For example, you can grant an administrator access to a single content repository folder and restrict administrative rights to only publishing content (not editing or deleting content).

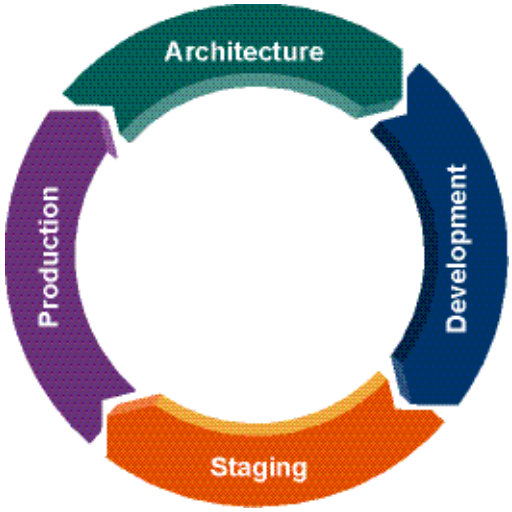
- **Content Management** - WebLogic Portal's virtual content repository lets you federate multiple compatible content management systems into a single interface, where you can centrally manage content and provide dynamic access to it in your portal applications. WebLogic Portal also provides its own content repository.
- **User Management** - In addition to an LDAP user store and authentication provider provided by Oracle, you can use third-party user stores with WebLogic Platform for authentication or retrieval of user properties. WebLogic Portal's unified user profile framework lets you create and maintain user profile properties that are attached to users regardless of which user store their user accounts reside in. User profile properties are key components in features such as personalization and access control.
- **Interaction Management** - WebLogic Portal provides an interaction management framework that lets you build personalization and campaigns, track user interactions with your portals, and trigger custom actions when specific events occur. Interaction management lets you improve the visitor experience, increase adoption and loyalty, and achieve business goals with visitor interactions.
- **Communities** - Communities are WebLogic Portal desktops that let users with common goals and interests work together in—and administer—a web-based environment. Whether for specific events, work groups, partners, or for any other groups that need to share information, communities provide a dedicated, secure, self-managed place to collaborate.
- **Production Operations** - Because portals are typically developed in a team development environment, WebLogic Portal supports production operations with tools that allow you to manage the portal life cycle, including portal development, staging, and production environments.
- **Federation** - WebLogic Portal supports federated portals. A federated portal is a portal that includes remotely distributed resources, including remote portlets, books, and pages. These remote resources are collected and brought together at runtime to a portal application called a consumer, which presents the federated portal to end users. Unlike a non-federated, entirely local portal, in most cases, the individual remote parts of a federated portal can be maintained, updated, and released independently without redeploying the consumer portal in which they are surfaced.
- **Client-Side Development** - Disc (Dynamic Interface SCripting) is a client-side JavaScript framework that handles the asynchronous updating of portlets, portlet events, and the retrieval of portal context objects. WebLogic Portal provides a set of web-based, REST-style APIs for retrieving, modifying, creating and updating portal data dynamically from the client.
- **Portlet Publishing** - Portlet Publishing makes portlets available to any web application over HTTP and allows portlets to be surfaced in any web page. With Portlet Publishing, for example, you can render portlets within a Struts or Spring application, or any other non-portal web page.

2.4 The Portal Life Cycle

Creation of a portal application with WebLogic Portal and other WebLogic Platform components follows an iterative life cycle through different environments: you design the portal application and build its foundation J2EE resources, develop the application functionality and user interface, perform runtime configuration and testing in a staging environment, deploy the application, and manage the application in a production environment.

Figure 2-2 shows the environments in which the portal life cycle occurs: architecture, development, staging, and production.

Figure 2 The Portal Life Cycle



2.4.1 Architecture

In most cases, the architecture phase occurs once. In the architecture phase, you make fundamental application design decisions and create a starting set of J2EE resources that serve as the foundation of your development efforts.

Resources you create at this phase may include a basic portal enterprise application and basic Web applications, custom EJBs and servlets (business logic), custom SQL scripts for pre-populating data in the development environment, shared development domain (`config.xml`) with string substitution variables, custom domain, enterprise application, and Web application templates, and custom look and feel resources. The architecture phase is where you set up your team development environment.

Application-level decisions you might make at this phase include the authentication and authorization providers you want to use, whether you will store user properties externally or use WebLogic Portal's user profile functionality to store user properties in WebLogic Server's default LDAP store (or both), which database you will use, your cluster configuration in staging and production, your source control strategy and conventions for concurrent application development, your process for moving application code and data between environments, whether or not there are remote books, pages, and portlets you want to consume in your portal application, and your content management strategy, especially if you are using a compatible third-party system in conjunction with WebLogic Portal's content repository.

In addition to application-level decisions, you must also make architecture and design decisions for each WebLogic Portal feature you will use. For example, if you want to

use campaigns in your application, what are the conditions that will trigger campaigns (such as date/time or user profile properties)? And when the campaigns run, what content characteristics to you want to use to display the content? Do you need a "season" property with values of "spring", "summer", "winter", and "fall", so that content added in the production environment with one of those values is picked up by campaigns automatically? There are many feature-level architectural decisions to make.

2.4.2 Development

In the development phase, developers use the set of resources created in the architecture phase to create programmatic portal application functionality. Multiple developers create application functionality using source control in this phase.

Resources you create at this phase may include JSPs, JSF applications, Web Services, portals, books, pages, portlets, campaigns, content selectors, placeholders, property sets, XML files, graphics, JavaScript, HTML, and Java code. These resources you create in the development phase are file based, and when you reach the production phase these resources are most likely deployed in an enterprise archive (`.ear`) file. The primary tools used in this phase are source control, Oracle Enterprise Pack for Eclipse and other editors of choice, and the WebLogic Portal production operations tools to move database and LDAP data between staging and development.

Because development code and resources are often dependent on server and database data, such as content, users, and security roles, development occurs in parallel with staging, where server and database data are populated.

2.4.3 Staging

In the staging phase, you assemble, configure, test, and deploy the portal application you want to go live with. With the testing aspect of the staging phase, there is tight iteration between staging and development until the application is ready to be released.

Tasks at this phase may include creating content and content types, creating desktops, modifying cache settings, creating administrative users, modifying interaction management rules, creating delegated administration roles and applying them to resources, creating visitor entitlement roles, connecting WebLogic Portal to external security and content providers, configuring services such as behavior tracking and the campaign e-mail, and modifying cache settings for performance. These tasks write data to the LDAP server and the database.

The primary tools used in this phase are the WebLogic Portal Administration Console, the WebLogic Portal production operations tools to move database and LDAP data between staging, development, and production, and any third-party content, authentication, or authorization providers you are using.

2.4.4 Production

In the production phase, you manage and fine tune your live portal application. Many tasks at this phase, such as user management, are identical to tasks you may perform in staging, such as adding users. But staging tasks are for the purpose of simply getting your public portals in place, and production is for true runtime application management.

Tasks you perform at this phase may include user management, applying of delegated administration and visitor entitlement roles to resources, creating desktops, adding content, and resetting/modifying interaction management rules. Visitors can also

create communities, as well as customize their portal environments using the WebLogic Portal visitor tools. These tasks write data to the LDAP server and the database. The primary tools used in this phase are the WebLogic Portal Administration Console, the WebLogic Portal production operations tools to move database and LDAP data between production and staging, and any third-party content, authentication, or authorization providers you are using.

When you are in production and you want to update the application functionality or user interface—create a new release of the application—you return to the development and staging phases until your updated application is ready to be released.

For ease of iterative development and application updates, WebLogic Portal provides production operation tools that let you move LDAP and database data between environments. For example, you can move portal desktops and communities from production to staging or production to development to test new application functionality against a live simulation of your production application.

Each feature area in WebLogic Portal follows the portal life cycle. Table 1 shows examples of how different features move through the portal life cycle.

Table 1 Examples of Feature Tasks in the Life Cycle Phases

Lifecycle Stage	Portals	Interaction Management	Security
Architecture	Design a custom portal look and feel and define the types of portals you want to build.	Determine the type of personalization you want to provide.	Develop a strategy for user access to your application.
Development	Create user interface graphics, cascading style sheets, JavaScript functions, skeleton JSP files, layouts, and portal, portlet, book, and page files; configure WSRP producer capabilities.	Create property sets with default values, campaigns, placeholders, content selectors; develop JSPs that implement campaigns, placeholders, and content selectors.	Configure declarative security in deployment descriptors to control access to EJBs, URLs, and file-based resources (such as JSPs); create property sets to support expression-based visitor entitlement and delegated administration roles.
Staging	Create portal desktops and community templates. Consume remote portlets.	Modify default user profile property set values and campaign, placeholder, and content selector definitions.	Create visitor entitlement and delegated administration roles, and apply those roles to portal resources for visitor and administrative access; create administrative users.
Production	Fine tune desktops and portal components. Users create communities.	Fine tune campaign, placeholder, and content selector definitions.	Fine tune role definitions and their application to portal resources; create administrative users and fine tune access rights for existing administrators.

Even though individual features follow the portal life cycle, development of the entire portal application and its features does not have to happen simultaneously. For example, you can move through the life cycle to create a complete portal application, then continue with additional portlet development that follows its own life cycle to update the entire portal application.

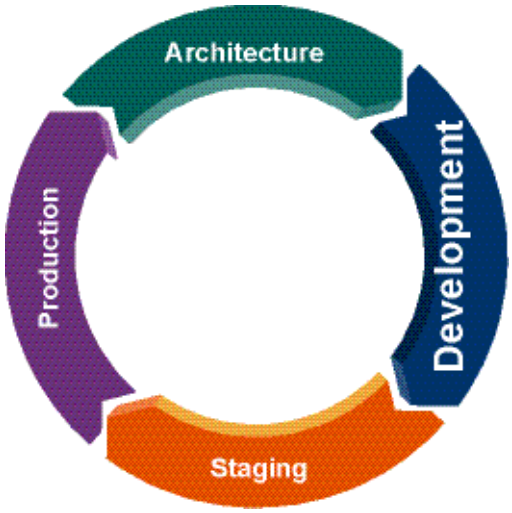
The remaining chapters in this guide discuss each phase of the portal life cycle in more detail and provide guidance on getting started with portal development:

- Portal Life Cycle: Architecture
- Portal Life Cycle: Development
- Portal Life Cycle: Staging
- Portal Life Cycle: Production

3 Development

Following the planning, design, and initial resource creation performed in the architecture phase, you can begin application development, as shown in Figure 3.

Figure 3 Development Phase of the Portal Life Cycle



In the development phase, developers use the set of resources created in the architecture phase to create programmatic portal application functionality and portal user interfaces, maintaining their work in a source control system. Resources you create in the development phase may include JSPs, JSF applications, web services, portals, books, pages, portlets, campaigns, content selectors, placeholders, property sets, XML files, graphics, JavaScript, HTML, and Java code (if you create EJBs, you are likely to do so in the architecture phase). These resources you create in the development phase are file based, and when you reach the production phase the resources are most likely deployed in an enterprise archive (`.ear`) file.

The primary tools used in this phase are source control, Oracle Enterprise Pack for Eclipse, other editors of choice, the WebLogic Portal Administration Console (for configuring database resources), and the WebLogic Portal Production Operations Tools to move database and LDAP data between staging and development, if necessary.

Figure 3 also shows the development phase overlapping with the staging phase. Because development code and resources are often dependent on database and security data, such as content, users, and security roles, development occurs in parallel with staging where database and security data are populated. While it is possible for developers to create database resources on their own development machines, setting up database resources in the staging environment provides a single location for all developers to share the same resources.

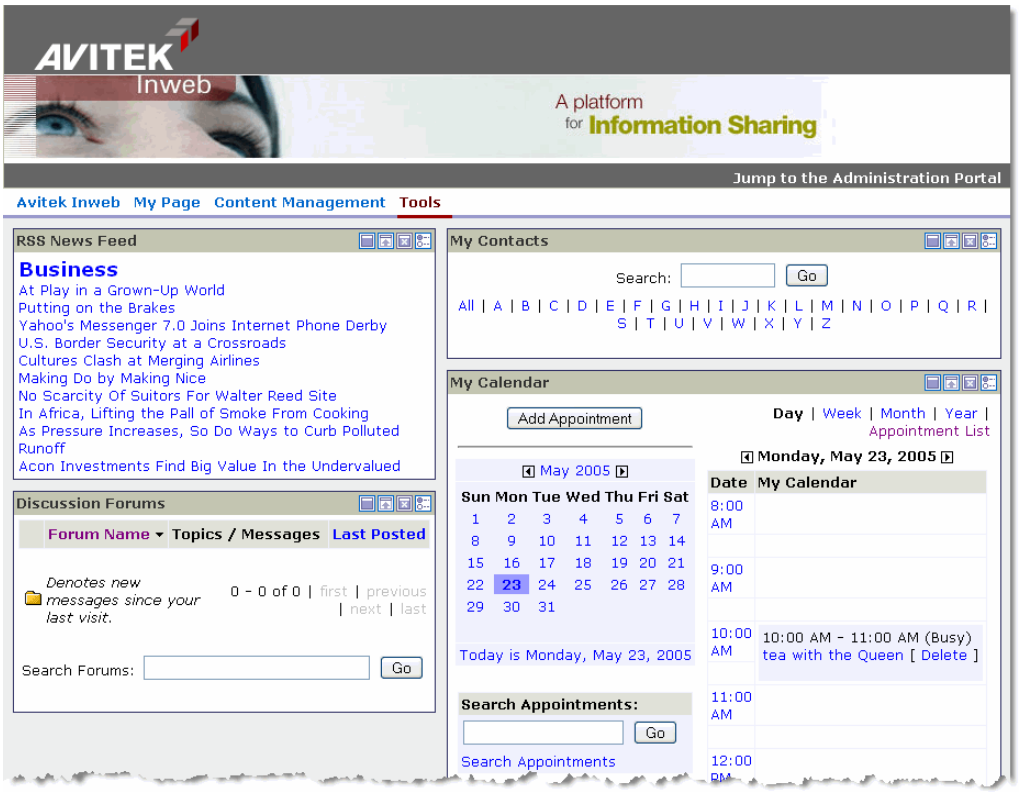
This chapter includes the following sections:

- Section 3.1, "Examples"
- Section 3.2, "Development Tools and Resources"

3.1 Examples

Figure 4 shows a sample portal desktop that includes a header region, multiple pages, and multiple portlets on the active page. The portlets contain various types of content and process flows, such as a news feed and a calendar.

Figure 4 A Portal Desktop



The following are possible development tasks that were involved in creating this portal desktop:

- Creating a portal file that uses a single-level menu of page tabs.
- Building a header JSP that uses a campaign to show a different graphic for different types of users. The header also includes a link that launches the WebLogic Portal Administration Console.

- Creating a look and feel, which includes JSPs that determine the physical structure of the layout, graphics, colors, font styles, and JavaScript functions that affect interface behavior, such as mouse overs on links.
- Developing the page flow applications that appear in each portlet. A page flow is made up of a Java file and JSPs. The Java file controls the flow of the JSPs that provide the user interface throughout the flow.

> **Note:** Page flows are a feature of Apache Beehive, which is an optional framework that you can integrate with WLP. See "Apache Beehive and Apache Struts Supported Configurations" in the *Oracle Fusion Middleware Portal Development Guide for Oracle WebLogic Portal*.

- Creating a JSF application that uses RSS to retrieve external news feeds from RSS providers.
- Creating portlet files to display the JSF application, then adding the portlets to the page. Developers can also retrieve remote books, pages, and portlets for inclusion in a portal.
- Adding help and edit functionality to the portlets. When clicked, the help and edit icons in the portlet title bars display help and show properties that can be used to modify the behavior or interface of the application. The icons link to JSPs that developers created to provide the functionality.

3.2 Development Tools and Resources

Developers use Oracle Enterprise Pack for Eclipse and other development tools for working with the majority of resources. Developers also use the WebLogic Portal Administration Console on their development domains to configure database, content management, and LDAP data for unit testing; although a best practice is to configure a central set of data in a staging environment that all developers can access.

The following are links to the development section for each of the WebLogic Portal feature guides:

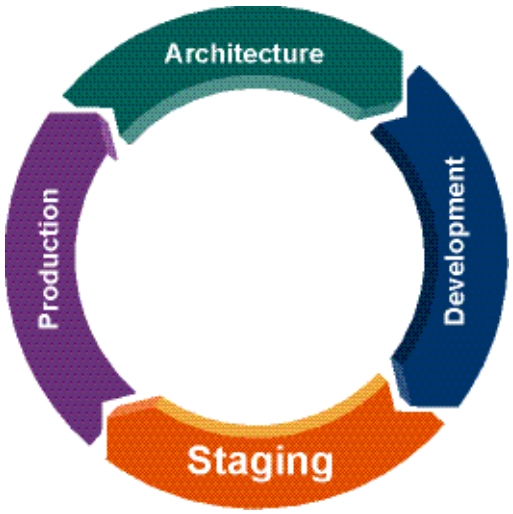
- *Oracle Fusion Middleware Portal Development Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Portlet Development Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Communities Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Content Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware User Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Interaction Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Security Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Autonomy Search Integration Sample Guide for Oracle WebLogic Portal*

See also the *Oracle Fusion Middleware Client-Side Developer's Guide for Oracle WebLogic Portal* for details on developing rich, interactive portlets.

4 Staging

While developers build portal functionality in the development phase, administrators and developers perform tasks at the staging phase, shown in Figure 5.

Figure 5 Staging Phase of the Portal Life Cycle



In the staging phase, you get your portal application ready for public consumption. You assemble, configure, and test the application, then deploy it to the production environment when it is ready for public consumption.

Using the WebLogic Portal Administration Console and resources created by the development team, you perform tasks such as creating desktops, creating desktop and community templates, creating content and content types, creating administrative users, creating delegated administration roles and applying them to resources, creating visitor entitlement roles and applying them to portal resources, modifying interaction management rules, connecting WebLogic Portal to external security and content providers, configuring services such as behavior tracking and campaign e-mail, and modifying cache settings to ensure optimum performance. You also test the application and deploy it to the production environment.

The primary tools used in this phase are the WebLogic Portal Administration Console, the WebLogic Portal Production Operations Tools (to move database and LDAP data between staging, development, and production), WebLogic Server application deployment tools, and any external content or security providers you are using.

Because development code and resources are often dependent on database and security data, such as content, users, and security roles, development occurs in parallel with staging where database and security data are populated. The staging environment provides a single location for all developers to share the same resources. The overlap between phases also allows iterative application testing.

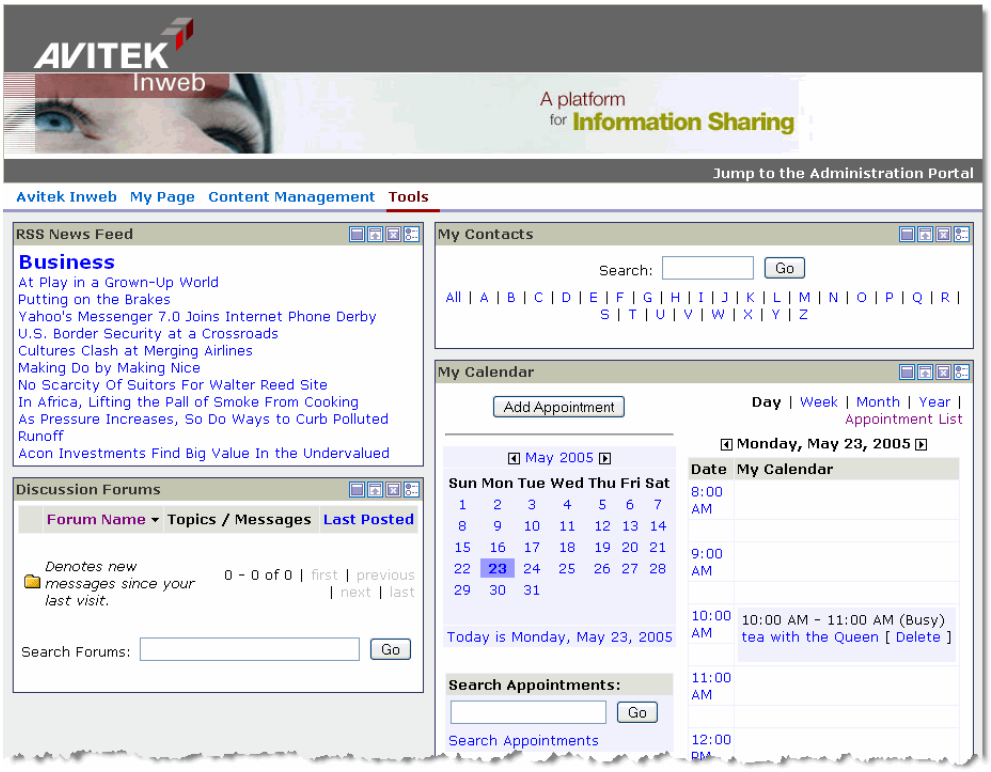
This chapter includes the following sections:

- Section 4.1, "Examples"
- Section 4.2, "Staging Tools and Resources"

4.1 Examples

Figure 6 shows a sample portal desktop that includes a header region, multiple pages, and multiple portlets on the active page. The portlets contain various types of content and process flows, such as a news feed and a calendar.

Figure 6 A Portal Desktop



The following are possible staging tasks that were involved in creating this portal desktop:

- Using the portal file created by the development team to create a desktop template. A portal administrators used the desktop template to create this desktop. If the desktop is to be a portal community, the portal administrator creates a community template from which end users can create communities.
- Setting entitlements on the Content Management page and its portlets so that only certain users can access them.
- Retrieving remote books, pages, and portlets for inclusion in the portal desktop.
- Modifying the query that retrieves content for the campaign used in the header.
- Creating content property types and applying those types to content added to the content management system.
- Creating administrative users that can modify the desktop, and setting up delegated administration so that specific administrators can administer only specific areas of the application.
- Modifying cache settings to ensure good performance.
- Configuring the campaign e-mail service.

- Configuring behavior tracking.
- Connecting WebLogic Portal to external authentication, authorization, and content providers.

4.2 Staging Tools and Resources

The primary tools that portal administrators and developers use in configuring, assembling, and deploying portal applications are the WebLogic Portal Administration Console and the WebLogic Portal Production Operations Tools.

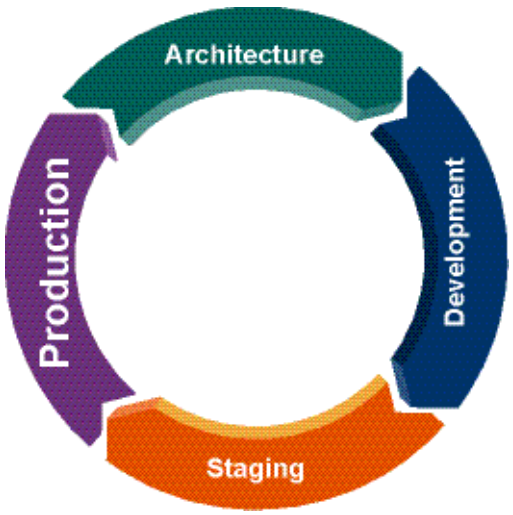
The following are links to the development section for each of the WebLogic Portal feature guides:

- *Oracle Fusion Middleware Portal Development Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Portlet Development Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Communities Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Content Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Interaction Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Security Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Autonomy Search Integration Sample Guide for Oracle WebLogic Portal*

5 Production

After you have assembled, configured, tested, and deployed your portal application, it moves into the production phase, shown in Figure 7.

Figure 7 Production Phase of the Portal Life Cycle



In the production phase, where users are accessing your application, you manage and tune your live portal application. Many tasks at this phase, such as managing users, managing content, modifying interaction management rules, and configuring visitor entitlements and delegated administration are identical to tasks you may perform in

staging. The difference is that the staging tasks are for the purpose of getting your public portals ready to go live, and production is for true runtime application management. Part of the portal life cycle involves moving live data from production back to staging for testing application updates against real data.

Tasks you perform at this phase include managing existing portal resources, creating desktops, creating desktop and community templates, managing users, applying of delegated administration and visitor entitlement roles to resources, adding and modifying content, and resetting or modifying interaction management rules.

In production, end users can also create communities, and they can customize their portal environments using the WebLogic Portal visitor tools.

The primary tools used in this phase are the WebLogic Portal Administration Console, the WebLogic Portal Production Operations Tools (to move database and LDAP data between staging, development, and production), and any external content or security providers you are using.

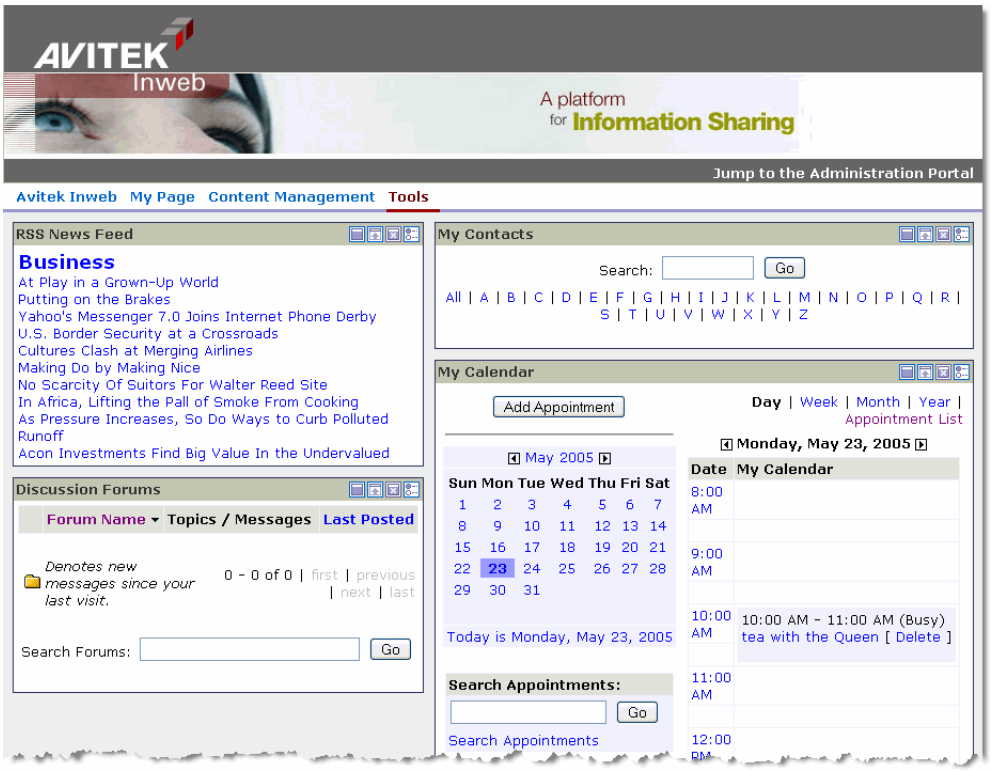
This chapter includes the following sections:

- Section 5.1, "Examples"
- Section 5.2, "Production Tools and Resources"

5.1 Examples

Figure 8 shows a sample portal desktop that includes a header region, multiple pages, and multiple portlets on the active page. The portlets contain various types of content and process flows, such as a news feed and a calendar.

Figure 8 A Portal Desktop



The following are possible production tasks that were involved in creating this portal desktop:

- Creating a desktop template that was used to create the portal desktop. If the administrator created a community template in the staging phase, an end user could have created this as a community desktop.
- Modifying the RSS News Feed portlet to point to a different news feed service. (The portlet provides edit functionality to let the user perform that task as well.)
- Modifying entitlements on the Content Management page and its portlets to expand the number of users that can access them.
- Retrieving remote books, pages, and portlets for inclusion in the portal desktop.
- Deactivating the campaign used in the header.
- Creating content property types and applying those types to content added to the content management system.
- Creating additional administrative users that can modify the desktop, and setting up delegated administration so that specific administrators can administer only specific areas of the application.
- Using the WebLogic Portal visitor tools, the user added a page called "My Page."

5.2 Production Tools and Resources

The primary tools that portal administrators use in managing portal applications in a live production environment are the WebLogic Portal Administration Console and the WebLogic Portal Production Operations Tools (for moving data among environments for development and testing (staging).

The following are links to the development section for each of the WebLogic Portal feature guides:

- *Oracle Fusion Middleware Portal Development Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Portlet Development Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Federated Portals Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Communities Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Content Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware User Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Interaction Management Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Security Guide for Oracle WebLogic Portal*
- *Oracle Fusion Middleware Autonomy Search Integration Sample Guide for Oracle WebLogic Portal*

6 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Oracle Fusion Middleware Overview for Oracle WebLogic Portal, 10*g* Release 3 (10.3.4)
E14241-02