BEA WebLogic Portal™

Architecture Guide

Version 9.2
Revised: June 2006

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

1. Architecting Portal Applications

CHAPTER 1

Architecting Portal Applications

Talk about the need to design your processes before you build your portals. (Per Josh’s roadmap slides.) Figure out how you’re going to surface your services (portal services, Java controls, Web services, data views). Web services? Content? Commerce? Search? After you figure that out, you can build your processes (workflows). Then build your UI (Page Flows, Crystal Reports), then distribute the portals (portal framework stuff: organize, federate, personalize, secure). There you have your process portal.

Domain – Contains server scripts and configuration, environment-setting scripts, database files and scripts, log files, default authenticator and role mapper ldift files, and other scripts and properties files.

Enterprise Application – Contains the WebLogic Portal API, controls, and portal datasync data.

Web Application – The visible part of your application. A Web application is a URL that end users can access. A Web application contains tag libraries, portal framework files, portals, portlets, and other resources such as JSPs and page flows that are surfaced in a portal interface.

Checklists

Use the following checklists to guide you through the design decisions you must make and to help you determine the resources you must create to support your portal development.

Domain

Multiple Enterprise Applications in a Domain

- Determine your staging strategy (such as cluster configuration, connection to an external authentication provider, and connection to an external content management system).
- Determine your production strategy (such as cluster configuration, connection to an external authentication provider, and connection to an external content management system).
- Determine whether you will store user properties externally or use WebLogic Platform's LDAP server.
- Determine which database you will use.
- Determine which external corporate systems you want to integrate into your portal applications.
- Determine your team development and source control strategies.
  - Create a shared development domain script that contains string substitutions for each developer machine.
  - Create a custom domain template for developers to create a development domain in their development environments.
  - Create custom SQL scripts for developers to populate their development databases.
  - Define coding conventions.
  - Define the process for moving application code and data between environments.

Portal Enterprise Application

- Create/reuse custom EJBs, servlets, Java source code, and other enterprise application resources.
- Create a custom portal enterprise application template for developers to populate their development environments.
- Design the types of corporate applications you want to develop (such as custom Administration Portals, executive dashboards, collaboration communities, ASP affiliate applications, ISV-branded applications, self-service applications, and OA&M applications.
- Determine security needs for each type of application you will create.
- Determine application cache settings for performance. These settings vary depending on the type of portal. For example, an outward-facing portal is likely to have different cache settings than an Intranet portal.

Portal Web Application

- Determine the desktops and community templates you want to create.
- Create custom Look & Feels.
- Create/reuse custom JSP tag libraries, Java source code, and other web application resources.
- Determine if there are remote books, pages, and portlets you want to consume in your portal application.
- Create a custom portal web application template for developers to populate their development environments.

Planning the types of portals you are going to build: internal, external high traffic, communities, etc. What are the performance and security requirements? e.g., If a big portal, focus on performance. Show a table of suggestions for most important considerations.

Also talk about the design/practical differences between B2E (employee), B2C, and B2B portals. Different traffic, purpose, number of users, different peak times, etc.

Determine where you'll store your Java source code for your portals. Discuss the diffrences between app-level and Web app-level Java code.

Point to the strategy section in the prodOps guide.

Design considerations, including using existing resources such as struts apps and Web apps.

Library Modules and Facets

Instructions on creating library modules and facets (or x-ref to where it lives)

Existing Applications

Provide conceptual guidance on how one would bring existing applications and projects in at the Architect stage. The actual steps should be covered in the Upgrade Guide.

Provide possible scenarios/flows for new or existing apps. For example, for someone with existing apps, here's how you'd merge into the life cycle and here's what it would mean to you. Or for a new app, here's what a flow might look like; like what kind of app do you want to build? Lead folks into the life cycle in a way they can understand.

Design Considerations

The complexity of a portal -- # of books, pages, portlets, buttons -- affects performance & scalability. When the portal is being assembled (from a file or the database), XML is parsed into a control tree and stored in memory. A portal served from database is cached to avoid going to the database after the portal is first built. If portal users have customized their portals, there is a cache entry for each user. The memory usage goes up as the number of users with customizations increase.

Remember the following when designing your portal:

- WebLogic Portal is a presentation layer
  - Keep business logic out of the presentation tier
  - Workshop controls provide a powerful abstraction layer
- Place items that require heavy processing in an edit page or a maximized URL
- Avoid large amounts of data retrieval in a portlet
- Avoid using too many entitlements, campaigns, and personalization in a single portlet or portal page.

Impact of Portlet Uses

As you add more pages, portlets, backing files and other portal controls ,the more execution time that is spent on the lifecycle methods as they walk the control tree. If you find yourself with more than 10 portlets on a page, rethink your design.

Using Forms in Portlets

Avoid using forms that update the data within the portlet; this causes the entire portal to refresh its data, which can be very time consuming.

x-ref Portal Solutions Catalog for goal:technology

The “Cost” of Backing Files

xx

Considerations for “Expensive Processing”

separate areas of expensive processing

Complicated Page Layouts

Intricate table layouts and other complex HTML can cause a perceived wait on the client after the files are transmitted and the browser determines how to render the page. Simple pages render more quickly.

Be mindful of the size of the response. Complex pages (e.g., lots of images in the page, header & footer) will produce a lot of html, and this must be downloaded by each visitor.

Image Content

More images require more downloading from the server, lengthening the time it takes for a Web page to complete rendering. The size of each image file is also a factor affecting performance. Although there may only be a few, large image files can slow down delivery of a Web page. Keep the number of images you use on a page to a minimum, and be sure those you do use are a reasonable size. Additionally, the location of those images from a network perspective is important. For instance, images should not need to traverse firewalls before arriving at a user’s browser.

SSL

The use of Secure Sockets Layer (SSL) in communication from a user’s Web browser to a server or from server to server can affect overall throughput. SSL should be used when encryption of sensitive data is required while in transit or when strong server authentication is required. However, SSL should not be used unless it is absolutely needed.

Index