AOT LAB

Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Advanced Software Engineering

JSR 168

Prof. Agostino Poggi

JSR 168

- Java Community Process:
  - http://www.jcp.org/en/jsr/detail?id=168
  - Led by Sun and IBM
  - 1.0 Final Release Oct 27, 2003.
- Its goal is to support interoperability between portlets and portals

JSR 168

- Defines a standard for vendor container independent portlet components
- Defines a set of APIs addressing areas of aggregation, personalisation, presentation and security
- Standardizes two main things:
  - How the portlet container manages portlet lifecycles
  - How the portlets are programmed

AOT LAB

JSR Portal Software Frameworks

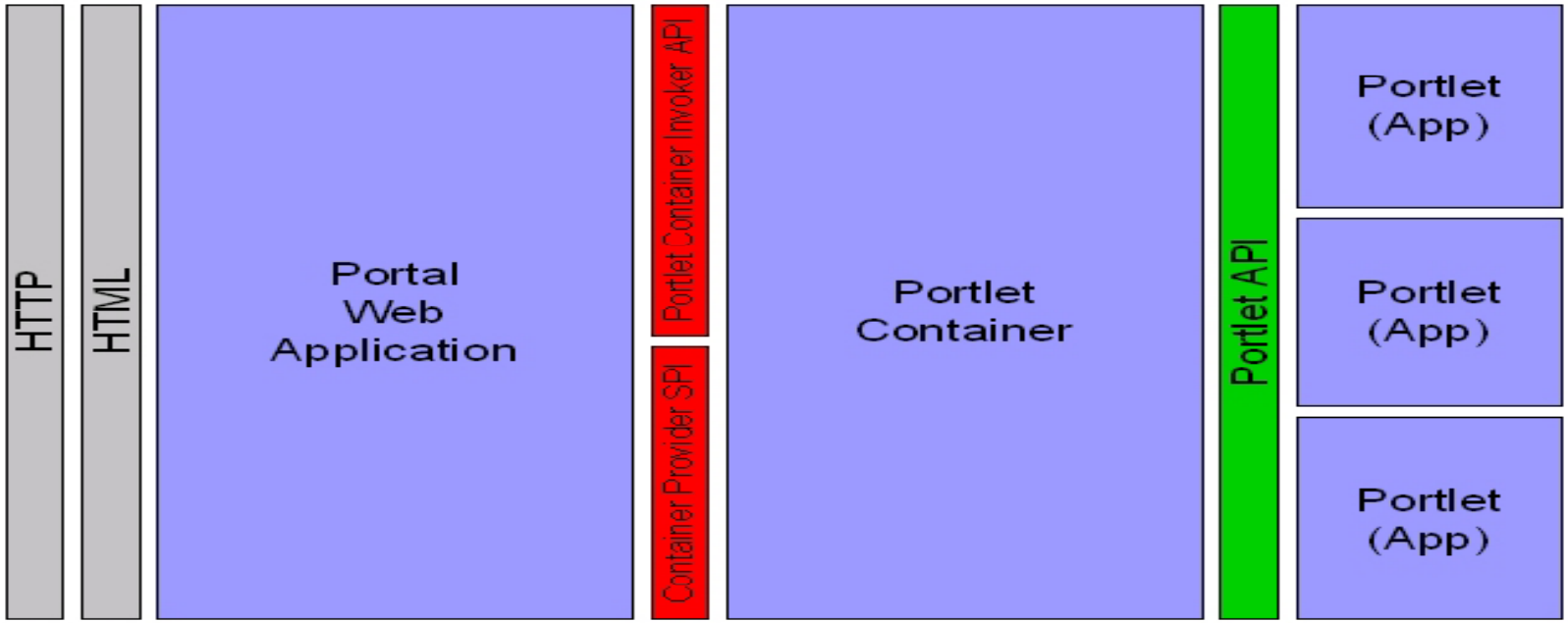
- Liferay
- JBoss Portal
- Apache JetSpeed 2
- Apache Pluto
- eXo platform
- uPortal
- IBM Websphere Portal
- BEA WebLogic Portal
- Sun Portal Server
- Oracle Portal

Portal

- A portal is a web application that commonly provides personalization, single sign on, content aggregation from different sources, and hosts the presentation layer of information systems
- Portal functionality are provided by:
  - A portlet container that manages the life cycle of the portlets
  - A content aggregator that aggregates the content generated by the portlet
  - A set of common services to simplify the use and development of a portal

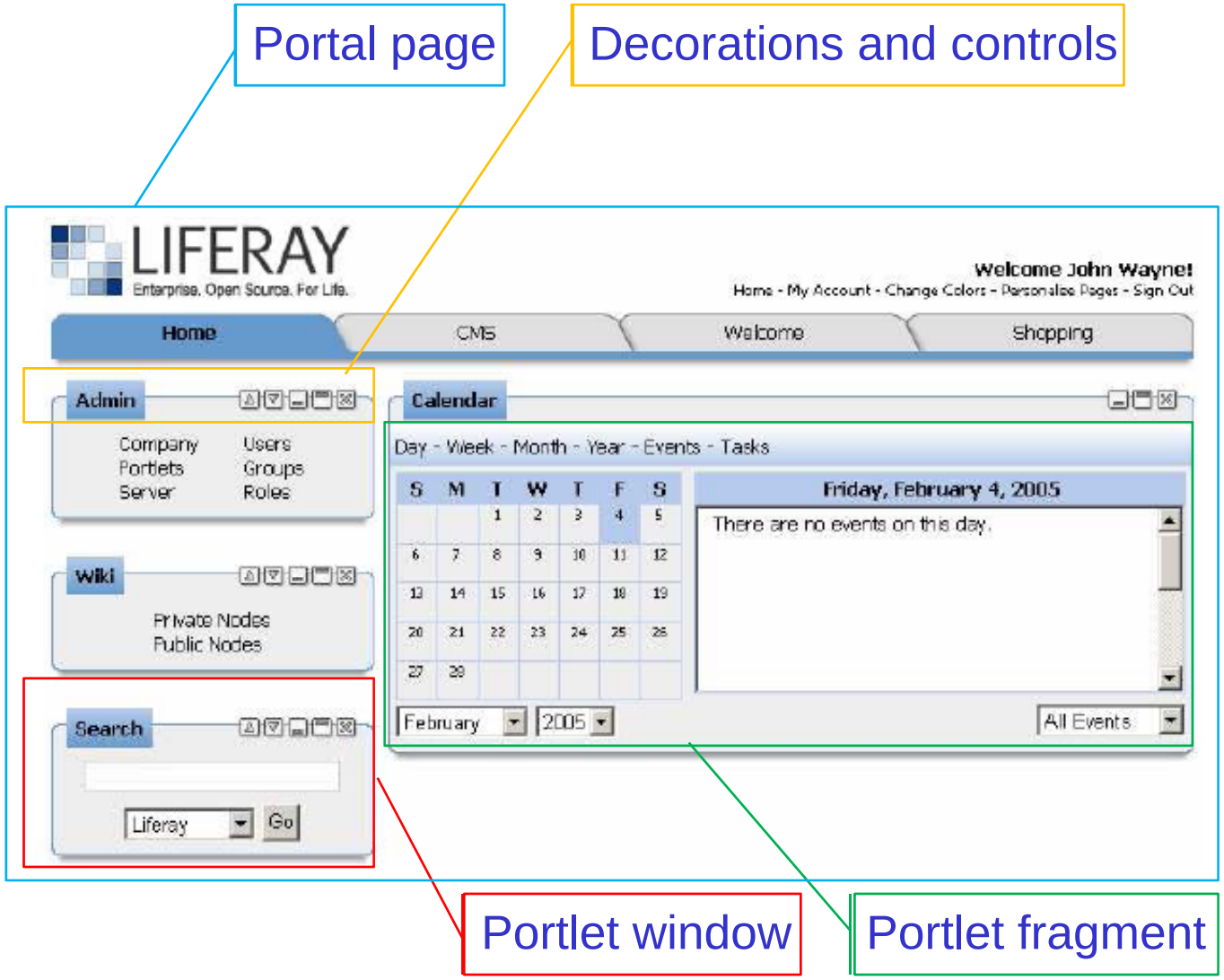
AOT LAB

Portal Architecture



AOT LAB

Content Aggregation



Portlet Container

- Extension of the Servlet Container
- Handles portlet components in addition to servlet and JSP components
- Shares much of the servlet container functionality (HTTP request handling, web application context, classloaders, session management, security)

Portlet

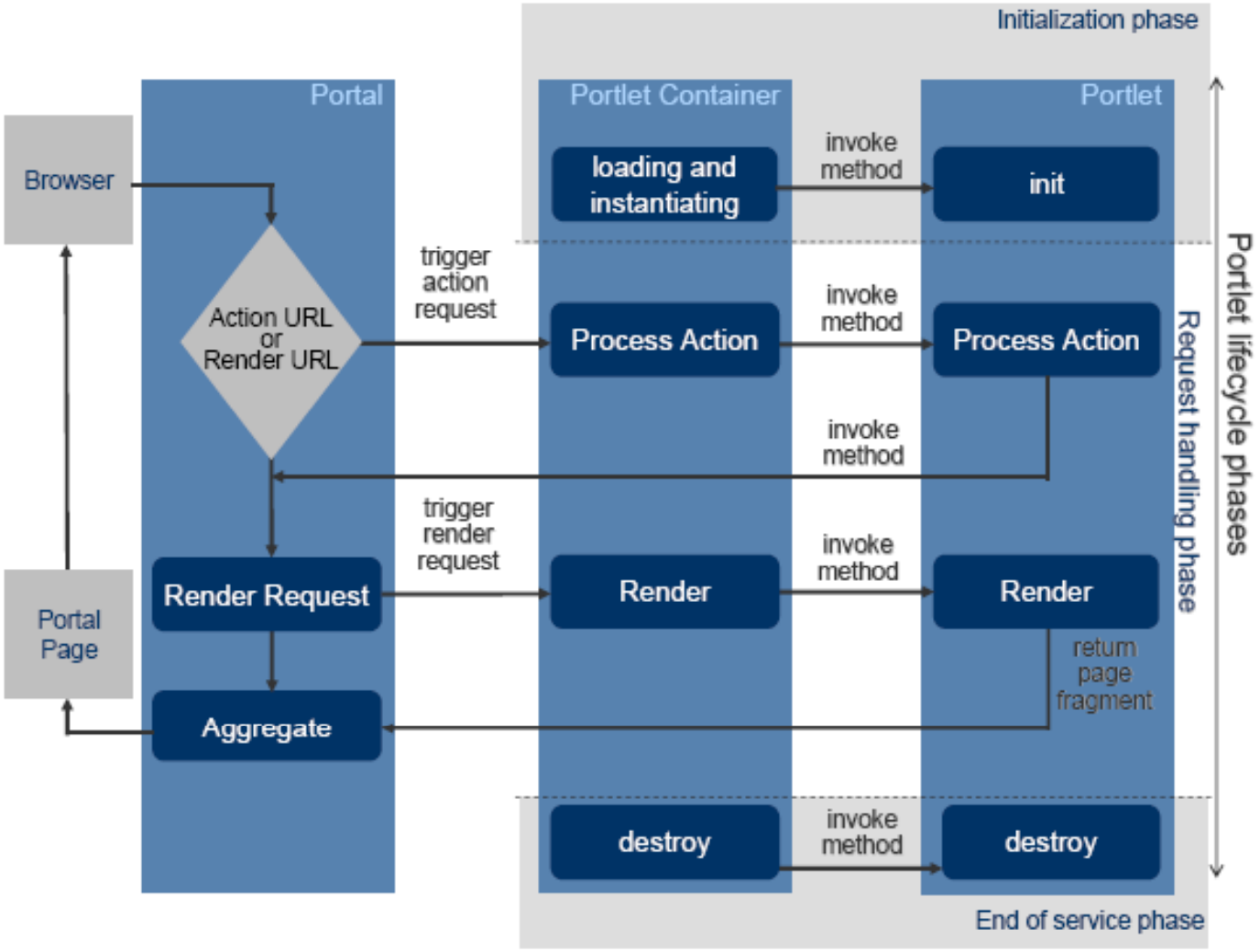
- A portlet is a piece of Java code that manages a piece of portal content usually called fragment
- A fragment is a piece of markup (e.g. HTML, XHTML, WML) adhering to certain rules and can be aggregated with other fragments to form a complete document
  - Normally is aggregated with the content of other portlets to form the portal page
- A portlet can do anything else that a Java Web application can do (e.g., access to a database, invoke a web service, download an RSS feed, ...)

Portlet Application

- A Web application that:
  - Groups portlets together with servlets and JSPs
  - Adds the portlet.xml deployment descriptor to the web.xml descriptor
- Portlets in the same portlet application share JARs, classes, and runtime stuff like request and session variables
- Portlets in different portlet applications do not share anything

AOT LAB

Portlet Life Cycle



Portlet

- Is a Java class that extends the GenericPortlet class
- Overrides some methods of the GenericPortlet class
- Implements the methods of the Portlet interface not already implemented by the GenericPortlet
- Uses some supporting classes and interfaces
  - Many are analogous to their servlet equivalents or are trivial wrapper around servlet equivalents

Portlet Interface Methods

- init
  - Called when the portlet is instantiated by the container
- processAction
  - Called after the user submits changes to a portlet
  - Intended to process input from a user action
- render
  - Called whenever the portlet is redrawn
- destroy
  - Called when the container destroys the portlet

GenericPortlet Class Methods

- Implements the render method and defines some specialized methods called by this method
  - doView
    - Called by render() when the portlet is in View mode. Intended to contain logic that displays the view page for the portlet
  - doEdit
    - Called by render() when the portlet is in Edit mode. Intended to contain logic that displays the edit page for the portlet
  - doHelp
    - Called by render() when the portlet is in Help mode. Intended to contain logic that displays the help page for the portlet

AOT LAB

Support Classes and Interfaces

- ActionRequest, ActionResponse
- PortletContext
- PortletMode
- PortletPreferences
- PortletRequestDispatcher
- PortletSession
- PortletURL
- RenderRequest,RenderResponse
- WindowState

Context and Session

- PortletContext
  - Similar to servlet context
  - Gets context info and the RequestDispatcher
- PortletSession
  - Stores attribute information for a single portlet application across multiple requests
- PortletURL
  - Creates URLs that reference the portal

Request, Response and Dispatcher

- RenderRequest and RenderResponse
  - Describe the request and response objects available to the render methods
  - Similar to the servlet request and response classes
- ActionRequest and ActionResponse
  - Describe the request and response objects available to the processAction method
  - Similar to the servlet request and response classes
- PortletRequestDispatcher
  - Includes and forwards to a JSP or a servlet in the same portlet

AOT LAB

Action Requests

- Do:
  - Invoke domain behavior
  - Change member data in domain (business) classes
  - Change portlet mode, window state, and portlet preferences
- Don’t:
  - Process rendering logic (e.g. paging)
  - Read data for display

AOT LAB

Render Requests

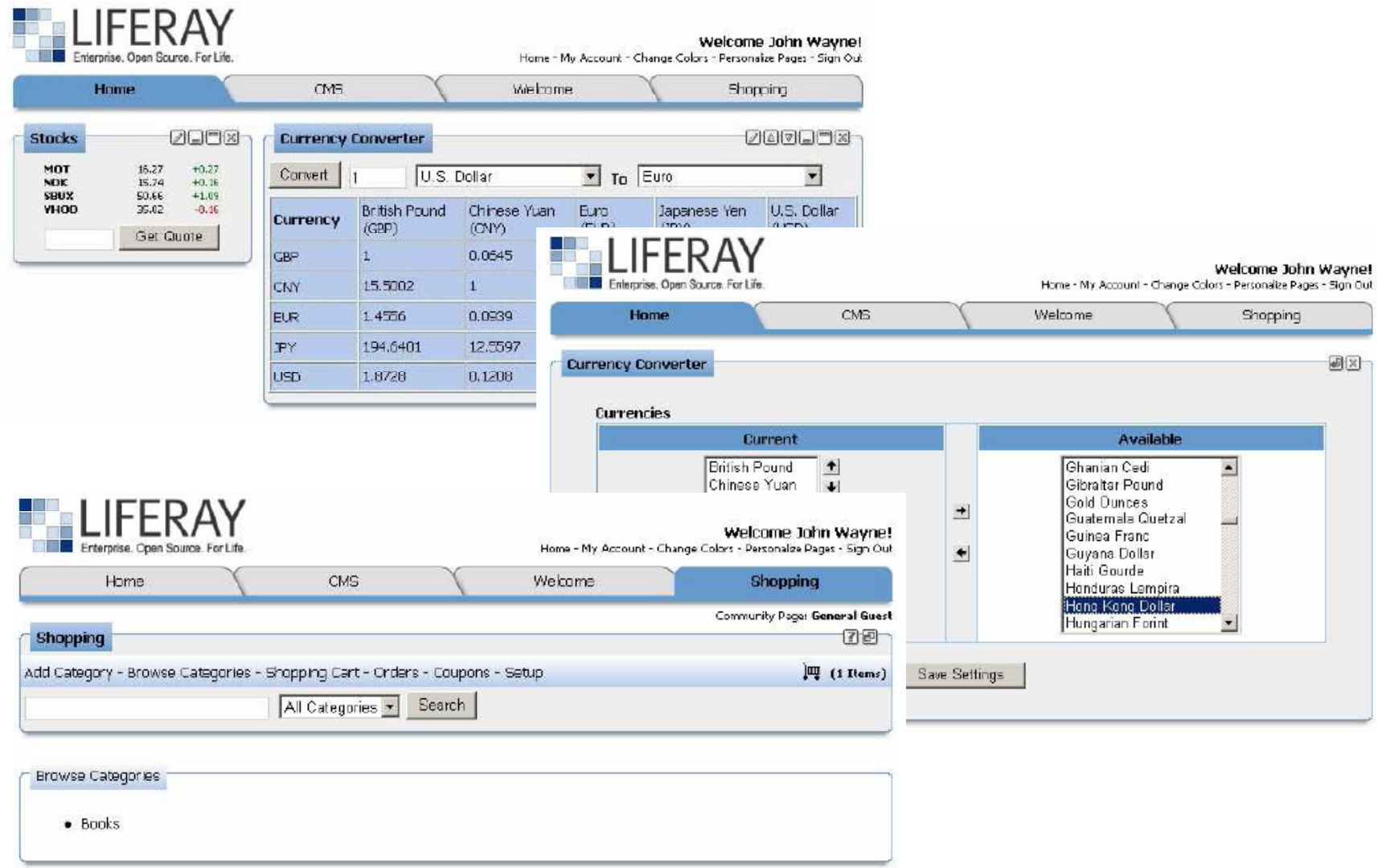
- Do:
  - Process rendering logic (e.g. paging)
  - Read data for display
- Don’t:
  - Invoke domain behavior
  - Change member data in model (business) classes
  - Change portlet mode, window state, and portlet preferences

Portlet Mode

- Portlet mode indicates the function the portlet is performing
- The possible portlet modes are:
  - View
  - Edit
  - Help
- These modes are used by the default render method to decide which lower level display method to call

AOT LAB

Portlet Mode

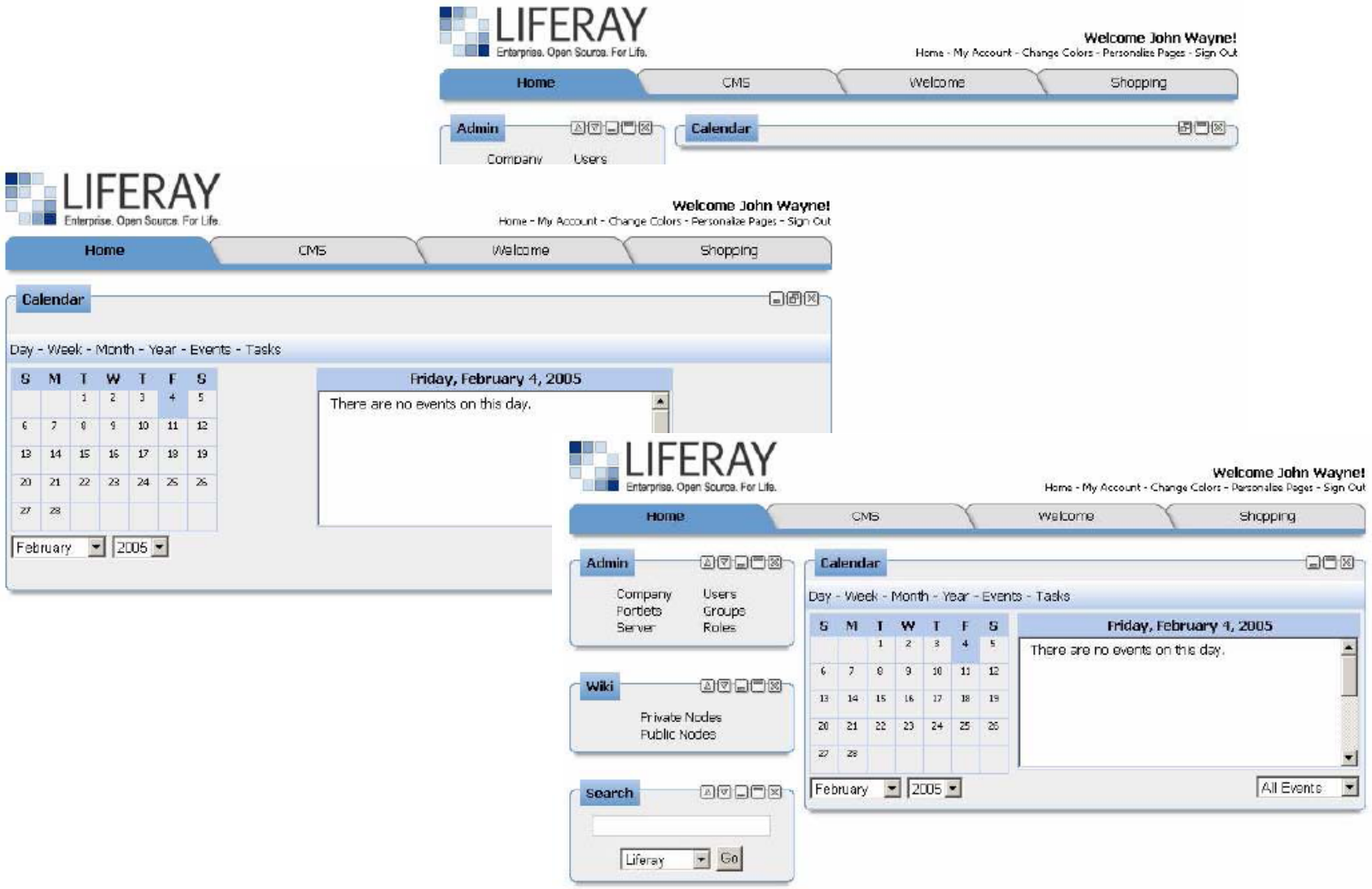


Window State

- Window state indicates the amount of portal page space that will be assigned to a portlet
- The portlet can use this information to decide how much information to render
- The possible window states are:
  - Minimized
  - Maximized
  - Normal

AOT LAB

Window State



Preferences

- Persistent portlet configuration managed by the portlet container
- Normally, portlet preferences are per portlet, per user
- Preferences can be read-only or read-write
- Default values are defined in the portlet.xml

Security

- Authentication is left to the underlying servlet container and authorization follows J2EE ‘roles’ model
- Programmatic role checking is managed through the isUserInRole method
- Transport security is managed in two ways:
  - Declarative in the deployment descriptor
  - Programmatic via PortletURL.setSecure()

Caching

- Portlets supports expiration based caching
- Default expiration time can be
  - Set in the deployment descriptor
  - Changed by portlets programmatically
- Cache is invalidated at expiration time or when the portlet is the target of an action

Portlet Tag Library

- Use of the tag library

```
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
```

- defineObjects tag
  - renderRequest, renderResponse, portletConfig

```
<portlet:defineObjects/>
<% renderResponse.setTitle("my portlet title"); %>
```

- actionURL and render tag
  - windowState, portletMode, var, secure

```
<portlet:actionURL windowState=”maximized” portletMode=”edit”>
```

- param tag

```
<portlet:param name=”employeeName” value=”DeniseSmith”/>
```

Portlet Application Deployment

- All resources, portlets and the deployment descriptors are packaged together in one web application archive (WAR file)
- There are two deployment descriptors:
  - One to specify the web application resources (web.xml)
  - One to specify the portlet resources (portlet.xml)
- All web resources that are not portlets must be specified in the web.xml
- All portlets and portlet related settings must be specified in the portlet.xml

Portlet Code Samples

- Liferay provides the source code of all its portlets
- Some of them are simple samples of use of a specific technology
- See:
  - http://www.liferay.com/web/guest/downloads/official_plugins