



Agent and Object Technology Lab
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma



Advanced Software Engineering

JSR 168

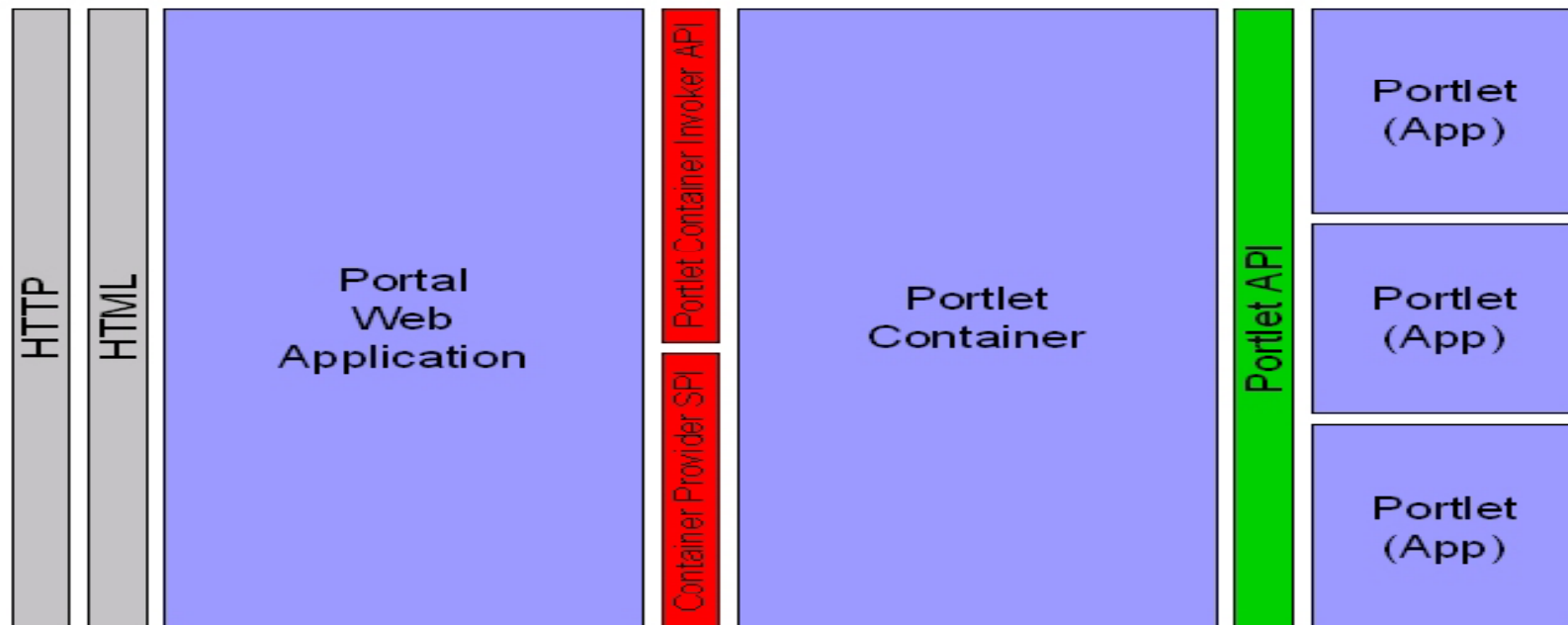
Prof. Agostino Poggi

- ◆ 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

- ◆ 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

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



The screenshot shows the Liferay portal interface. The top navigation bar includes the Liferay logo, the text "Enterprise. Open Source. For Life.", and a user welcome message "Welcome John Wayne!" with links for "Home - My Account - Change Colors - Personalize Pages - Sign Out". Below this is a secondary navigation bar with tabs for "Home", "CMS", "Welcome", and "Shopping".

On the left side, there are three portlet windows:

- Admin** (yellow box): Contains links for "Company", "Portlets", "Server", "Users", "Groups", and "Roles".
- Wiki** (blue box): Contains links for "Private Nodes" and "Public Nodes".
- Search** (red box): Contains a search input field, a dropdown menu set to "Liferay", and a "Go" button.

On the right side, there is a **Calendar** portlet window (green box) showing a calendar for February 2005. The selected date is Friday, February 4, 2005, with the message "There are no events on this day." Below the calendar is a dropdown menu for "All Events".

Annotations with colored boxes and lines point to specific parts of the interface:

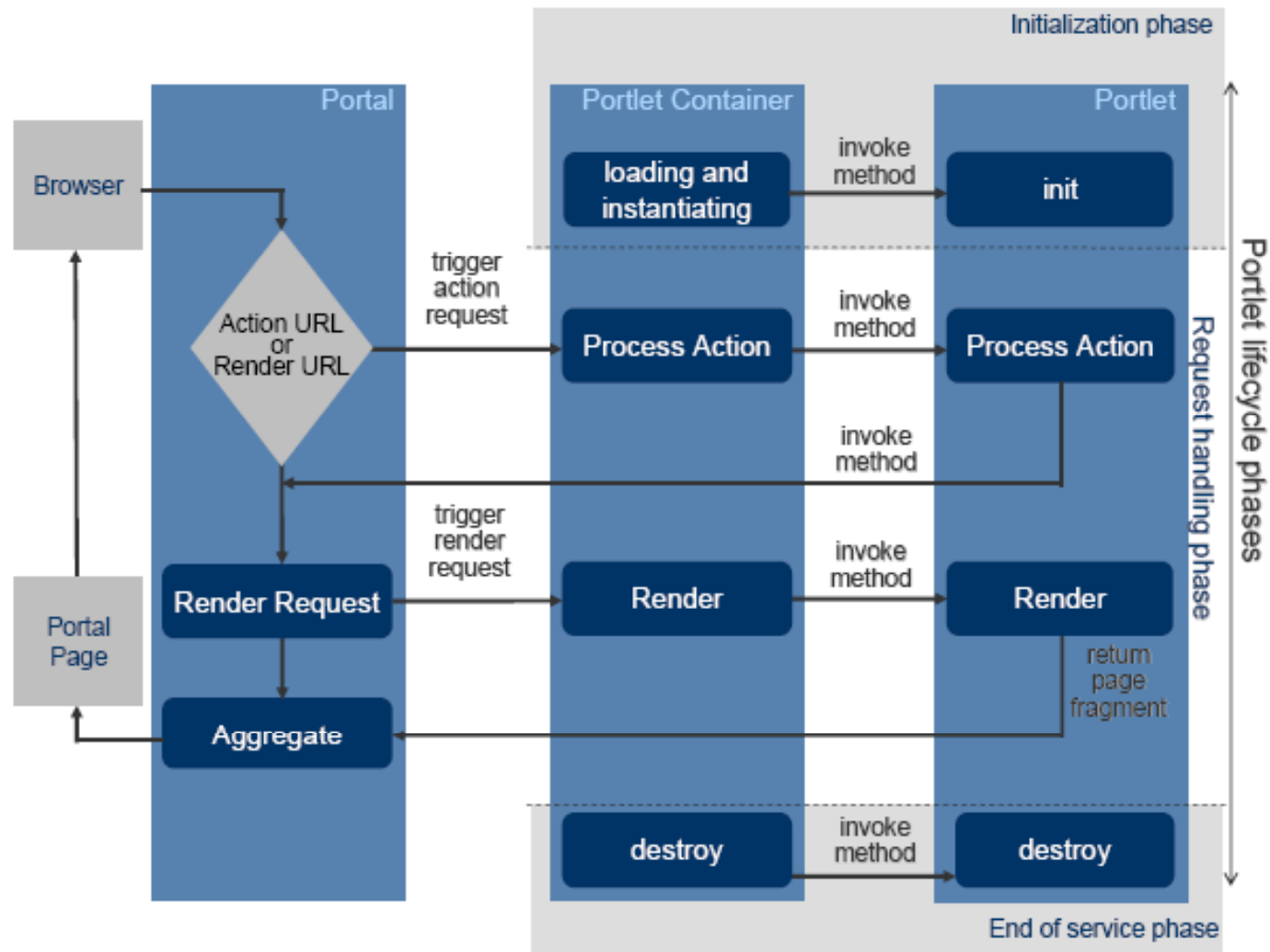
- Portal page** (blue box) points to the overall page layout.
- Decorations and controls** (yellow box) points to the top navigation bar.
- Portlet window** (red box) points to the Search portlet window.
- Portlet fragment** (green box) points to the Calendar portlet window.

- ◆ 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)

- ◆ 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, ...)

- ◆ 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

Portlet Life Cycle



- ◆ 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

- ◆ `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

- ◆ 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

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

- ◆ 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

- ◆ **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

◆ 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

- ◆ 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 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

The screenshot displays the Liferay Portal interface in Portlet Mode. The top navigation bar includes links for Home, CMS, Welcome, and Shopping. The main content area is divided into several portlets:

- Stocks Portlet:** Displays a table of stock prices for MOT, NOK, SBUX, and WHOD, along with a 'Get Quote' button.
- Currency Converter Portlet (Top):** A portlet for converting currencies. It shows a table of exchange rates for various currencies including British Pound (GBP), Chinese Yuan (CNY), Euro (EUR), Japanese Yen (JPY), and U.S. Dollar (USD).
- Currency Converter Portlet (Bottom):** A portlet for selecting currencies. It features a 'Current' list with British Pound and Chinese Yuan, and an 'Available' list with various international currencies like Ghanaian Cedi, Gibraltar Pound, Gold Ounces, etc.
- Shopping Portlet:** A portlet for browsing categories and shopping. It includes a search bar, a 'Browse Categories' section with a link to 'Books', and a 'Shopping Cart' section showing 1 item.

The Liferay logo and tagline 'Enterprise. Open Source. For Life.' are visible in the top left of each portlet instance. The user 'John Wayne' is logged in, and the interface is personalized with links like 'Home - My Account - Change Colors - Personalize Pages - Sign Out'.

- ◆ 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

The image displays three screenshots of the Liferay portal interface, illustrating different window states and layouts.

Top Screenshot: Shows the Liferay logo and navigation bar. The navigation bar includes links for Home, CMS, Welcome, and Shopping. The Admin menu is expanded, showing options for Company, Users, and Calendar. The user is logged in as John Wayne.

Middle Screenshot: Shows the Liferay logo and navigation bar. The Admin menu is expanded, showing options for Company, Users, and Calendar. The user is logged in as John Wayne.

Bottom Screenshot: Shows the Liferay logo and navigation bar. The Admin menu is expanded, showing options for Company, Users, and Calendar. The user is logged in as John Wayne.

The interface includes a calendar widget showing the current date (Friday, February 4, 2005) and a message stating "There are no events on this day." The calendar is set to February 2005.

- ◆ 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

- ◆ 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()`

- ◆ 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

- ◆ 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"/>
```

- ◆ 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

- ◆ 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