Portals@Apache

Standards and the Portals Project

Carsten Ziegeler
cziegeler@apache.org

Competence Center Open Source
S&N AG, Germany

About

- Member of the Apache Software Foundation
- Committer in some Apache Projects
  - Cocoon, Excalibur, Pluto, WSRP4J, Incubator
  - PMC: Cocoon, Incubator, Portals
- Chief Architect of the Competence Center Open Source, S&N AG, Germany
- Article/Book Author, Technical Reviewer
- Member of the JSR 286 spec group (Portlet API 2.0)

Agenda

- Portal Basics
- JSR 168
- WSRP
- Apache Portals Project

The Past (before JSR 168 and WSRP)

- Different Portal Vendors with their own APIs
  - No interoperability between local portlets and portal servers
  - Application and Content Providers must implement different portlets for different portal servers
- Quickly locked into a particular portal solution
- No standardized way to plug-n-play content and applications into portals
- No standardized way of integrating remote content

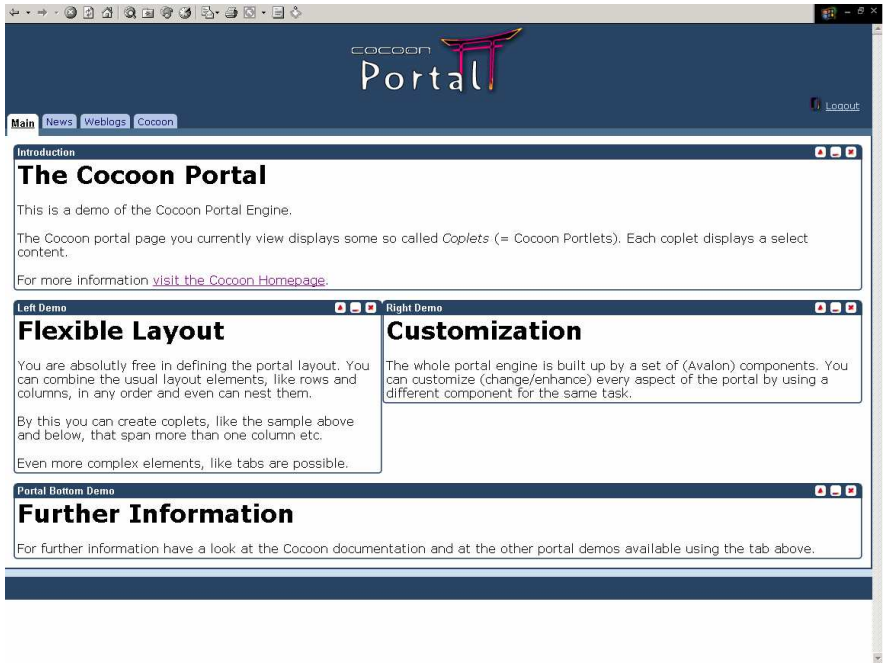
Portal Standards

Portals and the JSR 168

ApacheCon US 2006

What is a Portal?

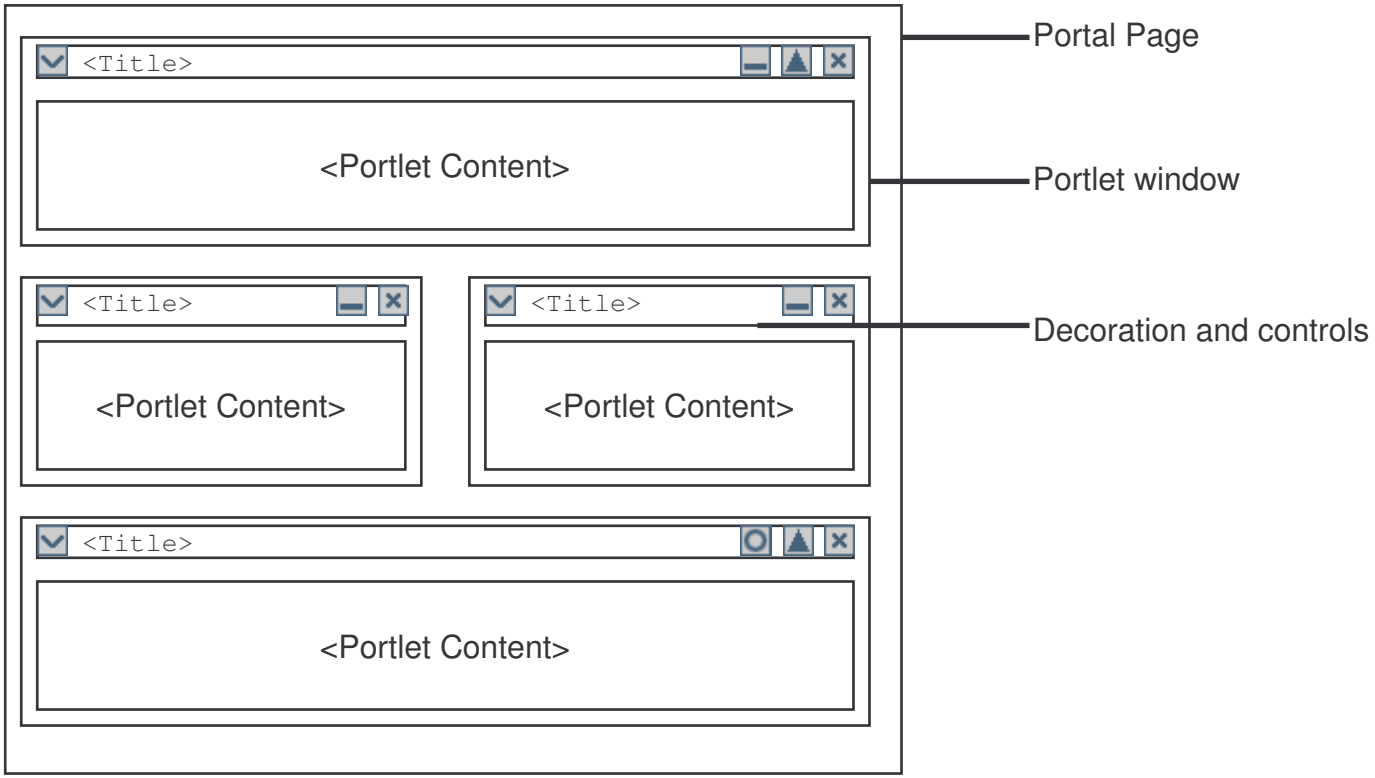
- Web Based Application
  - Personalization
  - Individualization
  - Content Aggregation
    - Using Portlets
  - Single Sign On



Common Requirements for a Portal

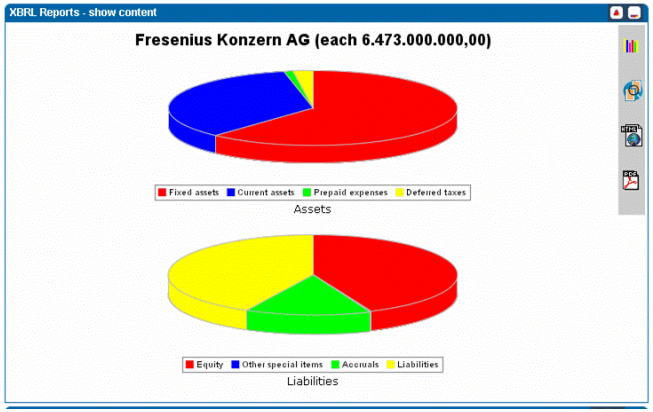
- Integration of different data sources
  - Static sources (HTML, XML, Office Documents…)
  - Dynamic sources (CMS, Archives…)
  - Databases (SQL DB, XML DB, LDAP…)
  - Complex Applications
- Multi Channel
  - PCs (HTML, XML)
  - Mobile, Organizer (WML)
  - Documents (PDF, Office Documents), Email
  - Applications

A Portal Page Sample

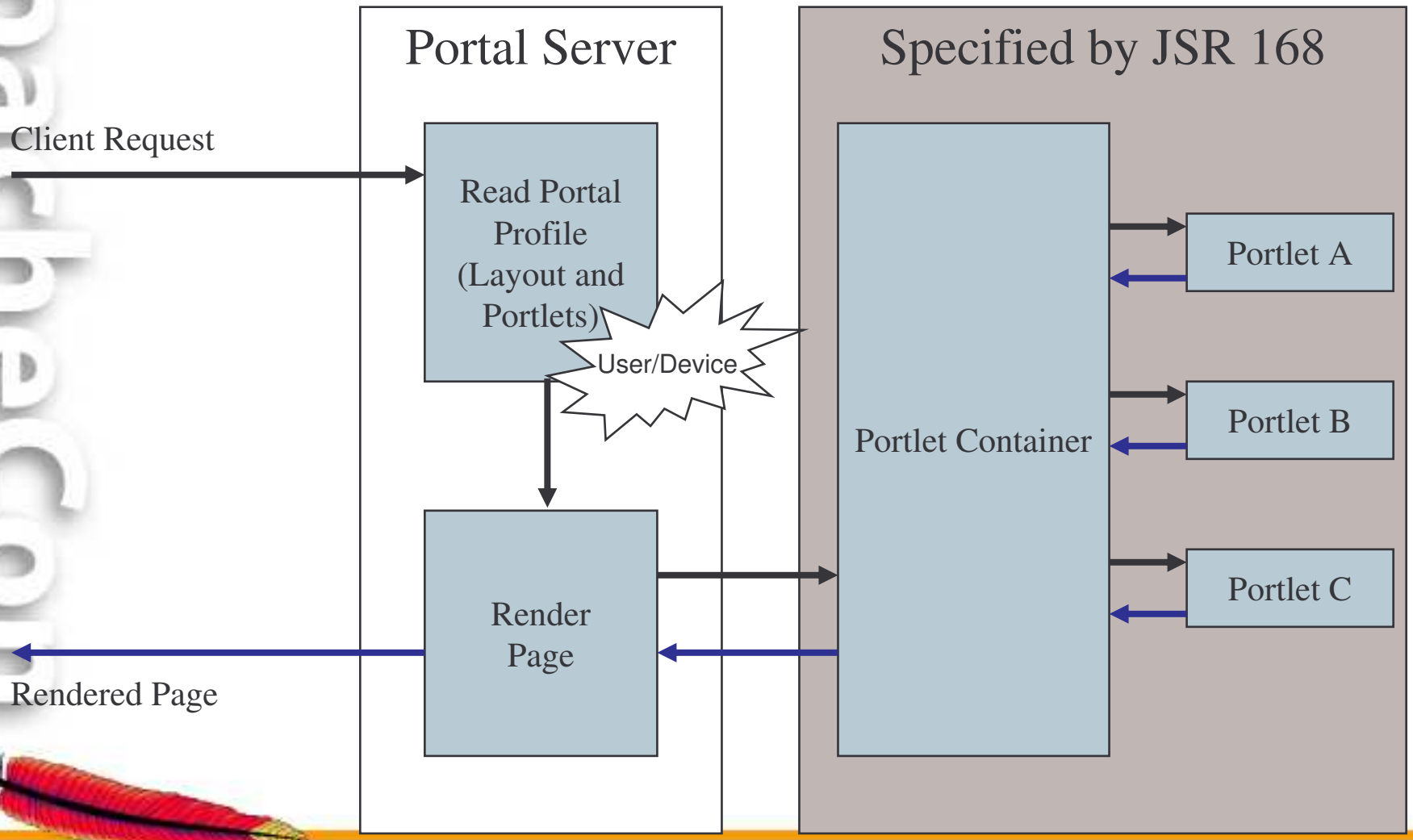


What is a Portlet?

- Web Component
  - Generates (dynamic) Content
    - News
    - Links
    - Complete Web Application
    - …
  - Managed by a Portlet Container



Overview


Portal Server
Specified by JSR 168
Client Request
Read Portal Profile (Layout and Portlets)
User/Device
Portlet Container
Portlet A
Portlet B
Portlet C
Render Page
Rendered Page

The JSR 168 – Portlet API

- Java API for interoperability among portlets and portals
  - Portlet Development (based on J2EE 1.3)
  - User Information and Preferences
  - Localization, Security
- Similar to Servlet API
  - Request-Response Cycle
  - Own Deployment Descriptor
- Portlet Container extends Servlet Container
  - Servlet Specification 2.3
  - **Not** covered in the JSR

Developing Portlets

- Write a Java class conforming to Portlet Interface
- Abstract class GenericPortlet can be used as basis
- Portlets are stateless wrt user (Singleton)
- Evaluation of portlet modes and window modes
- Generate content by writing into character stream
- Possible to use more sophisticated view layers:
  - JSP tag library is part of the specification
  - Different open source approaches
    - Bridges, JSF, Struts, Cocoon, Spring etc.

Portlet Life Cycle Methods I

Methods invoked once(!):

init(*configuration*)

- Instantiation by the container
- prepares the Portlet to serve requests

destroy()

- Destruction by the container
- cleans up the Portlet (no longer needed/shut down)

Portlet Life Cycle Methods

Methods invoked per "instance"/request:

processAction(*request and response*)

- Notification of changes/actions from the user
- process user input

render(*request and response*)

- Request to render the portlet in it's current state

Developing Portlets - Sample

```
public class HelloWorldPortlet implements Portlet  {

...

    public void render(RenderRequest req, RenderResponse res)
    throws PortletException, IOException {
        res.setContentType("text/html");
        Writer writer = res.getWriter();
        writer.write("<h1>Hello World</h1>\n");
        ...
    }
...
}
```

JSR 168 Elements

- Definition of valid markup fragments for
  - HTML / XHTML
  - CSS Styles
  - Namespacing
- URL Handling
- Portlet Lifecycle
- Modes and Window States
- (Caching)

Developing Portlets - Sample

```
public class HelloWorldPortlet implements Portlet  {

...

public void render(RenderRequest req, RenderResponse res)
throws PortletException, IOException {
   res.setContentType("text/html");
   Writer writer = res.getWriter();
   writer.write("<div class='portlet-font'>Hello
  World</div>\n");
   ...
}
...
```

CSS

Developing Portlets - Sample

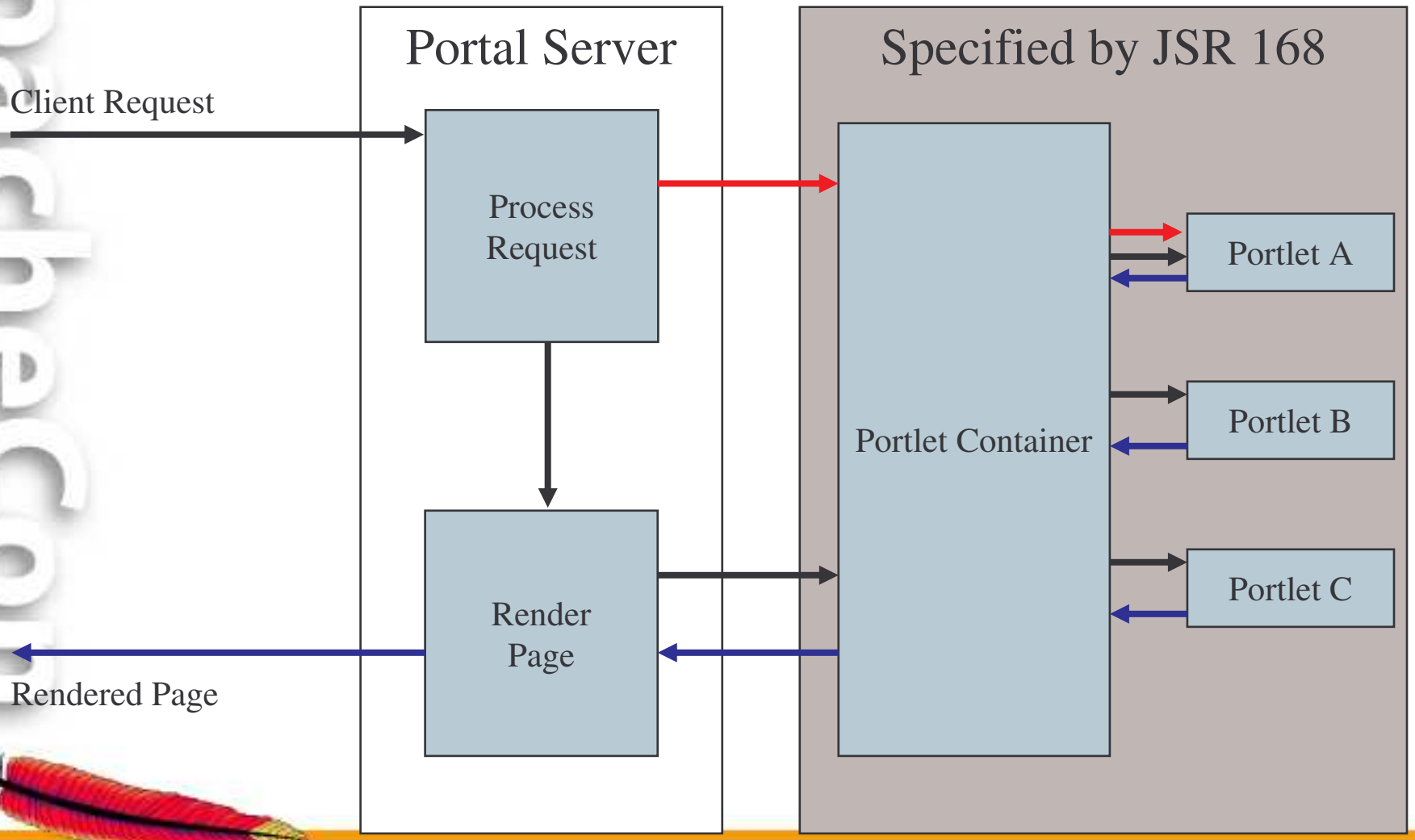
```
public class HelloWorldPortlet implements Portlet  {

 public void render(RenderRequest req, RenderResponse
  res)
 throws PortletException, IOException {
    ...
    PortletURL url;
    url = res.createRenderURL();
    url.setPortletMode(PortletMode.EDIT);
    writer.write("<a href='");
    writer.write(url.toString());
    writer.write("'>");
    writer.write("Edit mode");
    writer.write("</a>");
 }
}
```

Create a URL

User Interaction

- with content produced by portlets
  - Links or forms in the content
- with decoration
  - Links or buttons rendered by the portal
- Request/response cycle handled by the portal
  - Actions are forwarded to the portlets
  - Portlets may change their state
  - Page with all portlets is rendered

User Interaction - Flow



Portlet Modes

- Required
  - View – generate the content
- Optional
  - Edit – editing of user preferences
  - Help – provide help for the user
- Custom
  - About, Config, Edit_defaults, Preview and Print
- Portal vendor-specific modes are possible

Portlet Window States

- Required
  - Normal (default)
    - Portlet may share the view with other portlets
  - Maximized
    - Portlet has more space than usual
  - Minimized
    - Portlet should render minimal output/no output at all
- Portlet must handle all, but is free to generate the same content!
- Portal vendor-specific window states are possible

Portlet Preferences

- User specific data can be stored
- Service defined by the Portlet API
- Functionality provided by the Portlet container
- Access to preferences:
  - Action phase: read and write
  - Rendering phase: read only
- Default values in the deployment descriptor
- Preferences are key-value pairs
  - Value is either a string or an array of strings
  - Key is a string

Portlet Session Scope

- Portlet applications are Web applications
  - Sharing session with servlets
- Portlets can store private temporary data
  - Put with prefixes in the session (portlet scope)
- Portlets can share temporary data
  - Every component of the Web application can access it (application scope)
  - Sharing between: portlets, servlets, JSPs etc.

Portlet Deployment

- Portlets are deployed like a web application
  - war file
  - Including resources (images, JSP etc.)
- Two deployment descriptors
  - Web application
  - Portlet application (portlets, configuration)
- Portlet container may inject information into each Portlet application during deployment

Portlet Deployment Descriptor (Extract)

```
<portlet>
    <description>TestSuiteDescription</description>
    <portlet-name>TestPortlet1</portlet-name>
    <portlet-class>HelloWorldPortlet</portlet-class>
    <init-param>
        <name>config</name>
        <value>/WEB-INF/testsuite-config.xml</value>
    </init-param>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>VIEW</portlet-mode>
        <portlet-mode>EDIT</portlet-mode>
        <portlet-mode>HELP</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <supported-locale>de</supported-locale>
```

Advantages of the Portlet Specification

- Multiple Portal products can be supported
- Reusable code and portlets possible
  - More and more (open source) portlets are available
- Common tools are possible
- Open Source solutions available
- Rules for the markup (HTML with CSS, namespacing)

Potential Problems of the Portlet Spec.

- Important areas are not covered yet
  - Inter-Portlet communication
  - Potential danger of using vendor-specific features
  - Each portal solution provides add-ons
    - communication, services, component containers etc.
- Characters based approach
  - No direct XML Support

The Present (with JSR 168)

- Standardized API
  - Vendor specific add-ons
  - Quickly locked into a particular portal solution
- Bridges are used for implementation
  - Cocoon, JSF, Struts, Spring etc.
- Start using JSR 168
- Migrate only if required
- Integration of “complete” webapps as a portlet
  - Use generic proxy portlets
  - Or: WSRP

The Future (JSR 286)

- Portlet Specification 2.0
  - Started January 2006
  - First public draft soon available – scheduled for Q1/2007
- Corrections and clarifications
- Aligns with WSRP 2.0
- Add access to Composite Capability/Preference Profiles (CC/PP) data via the JSR188 API
- Introduction of portlet filters
- Inter-portlet communication
- Extended Lifecycle – notion of portlet instances
- J2EE 1.4 support
- Enhanced caching support

Portal Standards

Remote Portlet API - WSRP

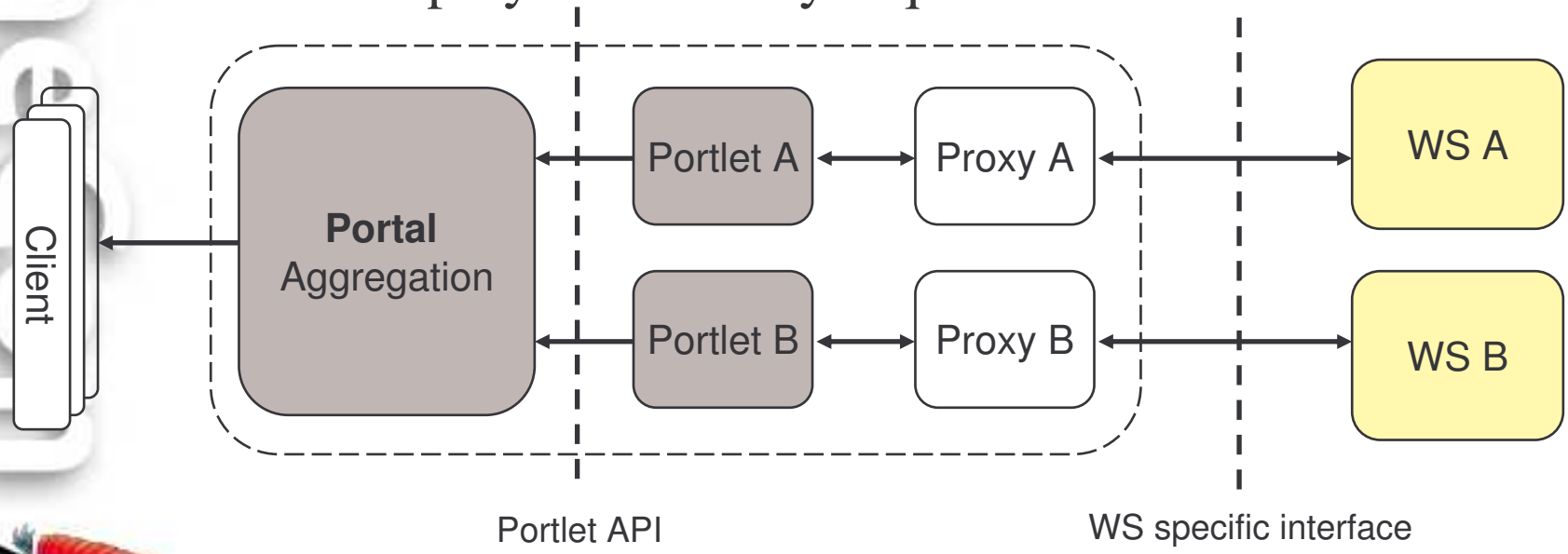
ApacheCon US 2006

WSRP–Web Service for Remote Portlets

- A standard for interactive, presentation-oriented web services
  - not tied to a programming language
  - publishing and consuming of content
- Sharing of portlets (markup fragments) over the internet with a common interface
- JSR 168 portlets run in the Portal Server – WSRP portlets run on a different server

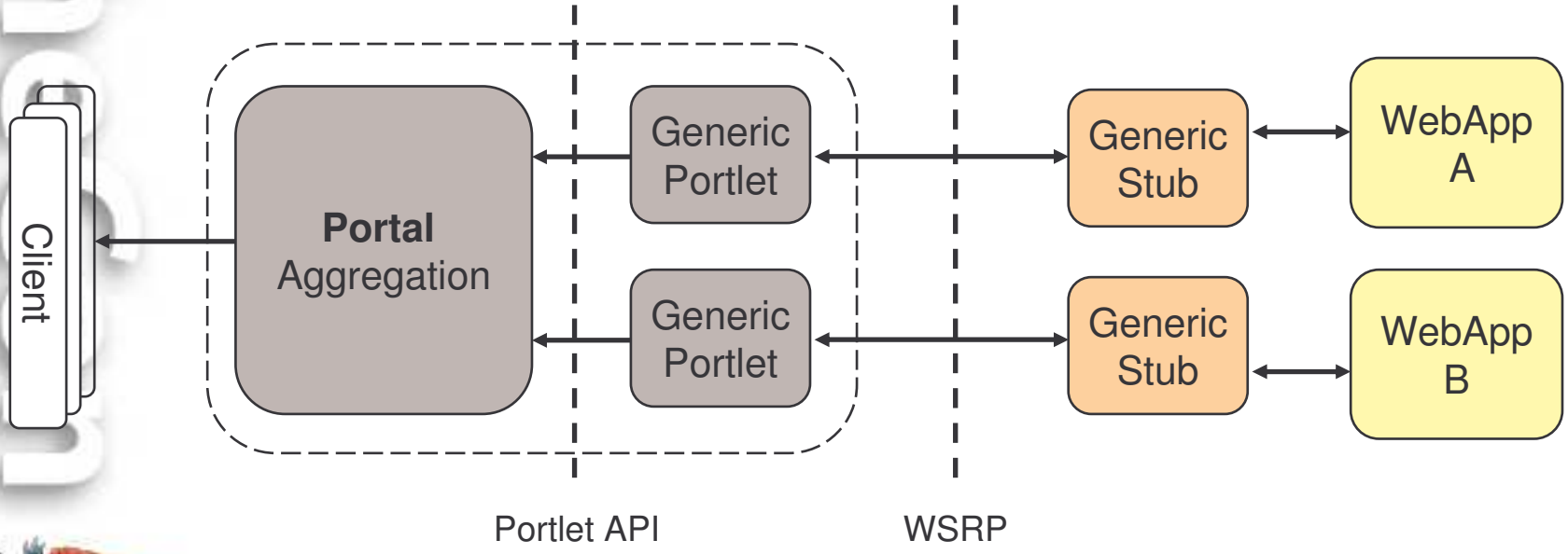
Portlets Using Web Services (Traditional)

- Different WS have different Interfaces
- Customized Proxies for each WS required
- Code/Deployment locally required

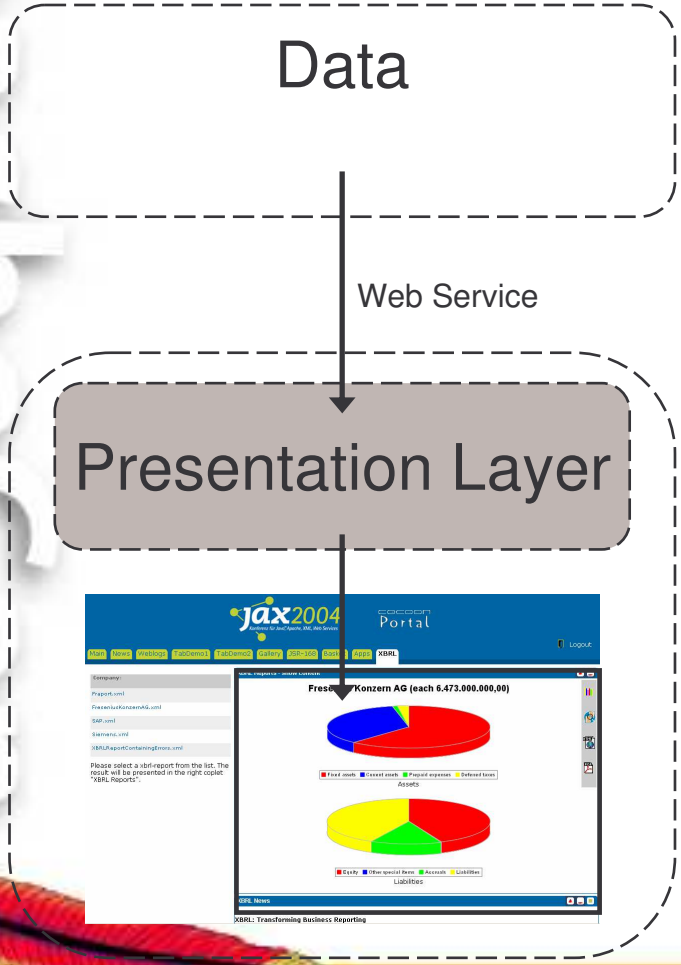


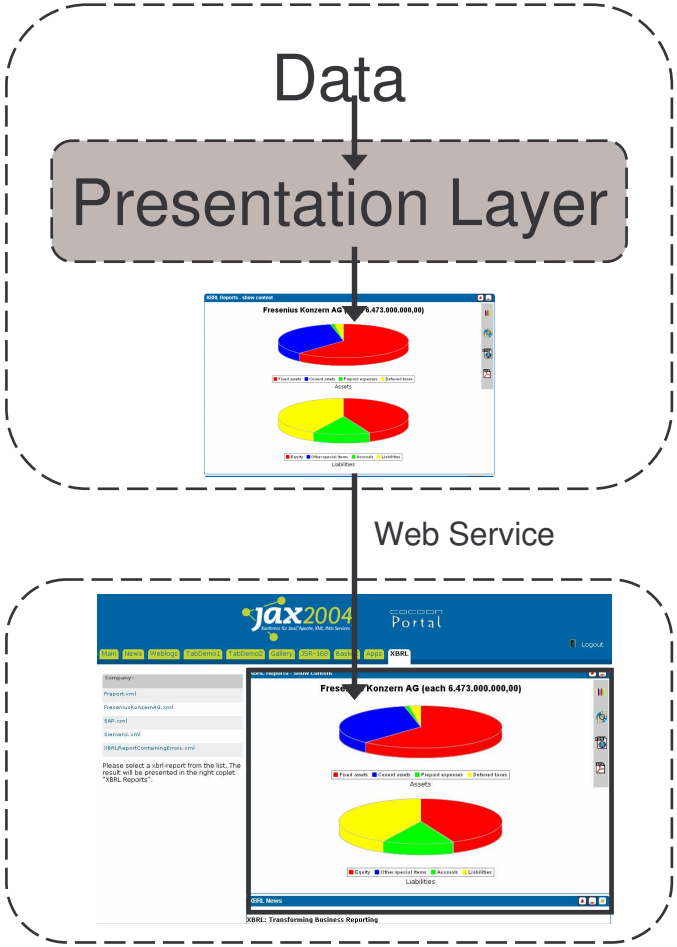
Portlets Using WSRP

- Unified API for WS
- No coding required: (available) generic code
- Presentation-oriented



Data Oriented Web Services vs. WSRP





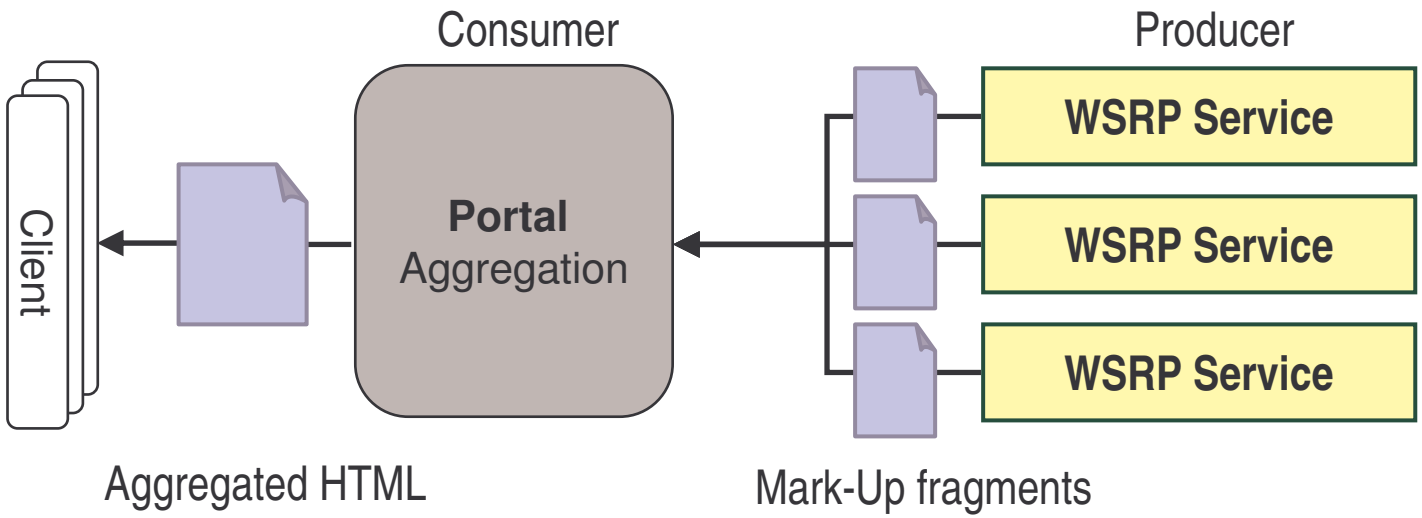
WSRP Elements

- Definition of valid markup fragments for
  - HTML / XHTML
  - CSS Styles
  - Namespacing
- URL Rewriting (Consumer and Producer)
- Session Handling
  - Context: User and device information
- Portlet Lifecycle
- Modes and Window States
  - View, edit, help, preview
  - normal, minimized, maximized
- Resource Proxying

WSRP Interface (WSDL)

- Service Description (mandatory)
  - Consumer queries Producer
- Markup (mandatory)
  - Getting content and user interaction
- Portlet Management (optional)
  - Consumer creates own customized instances
- Registration: (optional)
  - Consumer can register with Producers

Using WSRP in a Portal



- Portals can aggregate presentation from many WSRP services
- WSRP services can be aware of portal context
  - User profile from portal
  - Desired locale and markup-type
  - Active user agent

WSRP – Sample Markup Fragment

```
...
Click here on <A HREF="wsrp_rewrite?wsrp-
  urlType=blockingAction&wsrp-
  mode=wsrp:view&wsrp-
  interactionState=XXX&wsrp-
  windowState=wsrp:normal&wsrp-
  secureURL=false/wsrp_rewrite">Action</A>
  URL.
<B>Namespace: </B>
Pluto_127.0.0.1_1100620743364_2_someFunction
  Here()
<BR/>
...
```

Create a URL

Namespace

WSRP Achievements

- Plug&Play interoperability
  - between Content Providers and Portal Vendors
- Interoperable across a variety of WS stacks
- Markup retrieval, interaction processing
- Separation of Concerns
  - Security relies on underlying stack (WS-security, SSL)
  - Other concerns can be added, e.g. Billing
- Alignment with JSR 168

Advantages of WSRP

- Standardized way of integrating services
  - Plug&Play – generic components
- Services are already presentation oriented
- Common tools are possible
- Open Source solutions available
- Rules for the markup (HTML with CSS, namespacing)

Potential Problems of WSRP

- Not very common (today)
- A Step back to HTML
- Availability of own solution depends (additionally) on availability of all used services

Future of WSRP – 2.0

- Event Handling
- Additional markup types (VoiceXML, WML, cHTML)
- Add access to Composite Capability/Preference Profiles (CC/PP) data
- Enhanced Caching
- Attachments

JSR 168 and WSRP

- JSR 168 aligns closely with the WSRP
  - (JSR 286 and WSRP 2.0 will, too)
- Emerged at the same time
- Released open source implementations
- Both standards strive to work well together
  - Similar modes/functionality

Portal Standards

The Apache Portals Project

ApacheCon US 2006

The Apache Portals Projects

is a collaborative software development project dedicated to providing robust, full-featured, commercial-quality, and freely available **Portal related software** on a wide variety of platforms and programming languages. This project is managed in cooperation with various individuals worldwide (both independent and company-affiliated experts), who use the Internet to communicate, plan, and develop Portal software and related documentation.

The Apache Portals Projects

- Current Projects
  - Jetspeed 1/Jetspeed 2
  - Pluto
  - WSRP4J (Incubation)
  - Bridges
  - Graffito (Incubation)
- Related Projects
  - Apache Cocoon Portal

Apache Pluto

- Reference Implementation of the JSR 168
- Framework for building
  - A consumer (into a portal solution)
  - A provider (into a framework)
- Test harness
  - Startup Pluto and upload your portlets!

Apache WSRP4J (Incubation)

- Facilitate quick adoption of WSRP
- Framework for building
  - A consumer (into a portal solution)
  - A provider (into own application)
- Testing

Apache Portals Bridges

- Support for portlet development (JSR 168)
- Build a web app with your favorite framework
  - Struts, JSF, Velocity
- Use Portal Bridges to deploy this as a portlet
- Transparent portal integration not always possible
  - Follow the provided guidelines
- Version 1.0 is released

Apache Portals Graffito (Incubation)

- Framework to build content based apps
  - CMS, forums, blogs etc.
- Provides JSR 168 portlets
- Features
  - Taxonomy
  - content versioning, fine grained access control
  - collaborative editing, publication workflow
  - scheduling, indexing, searching and more ☺
- Support for many document types
  - like XML, HTML, PDF, Office

Apache Jetspeed 2

- Enterprise portal solution
  - Supports portlet standard (JSR 168)
  - Supports Portals Bridges
  - Component based
- SSO
- Flexible layout (XML description)
  - Template support
- Several usable portlets
  - Administration and User
- AJAX Support (Desktop 2.1)
- Final Version is out!

Apache Cocoon Portal

- Enterprise Portal Solution
  - Based on Apache Cocoon
  - Portal Framework to build portals
  - Supports portlet standards (JSR 168 and WSRP)
  - Supports Portals Bridges
  - Supports Cocoon Applications
  - Component based
- Flexible layout engine (XML/XSLT) – (AJAX in 2.2)
- Powerful Event Mechanism
  - Status changes
  - Portlet communication

Conclusion

- Current Portal Standards
  - Provide a good basis, but aren't covering all important parts, but will be extended
- Several Open Source Solutions
  - Apache Portals (and others)
  - Increasing development efforts (AJAX)
- Use standards **with** additional frameworks
  - E.g. JSR 168 with Spring MVC Portlet

Thanks for your attention!