WSRP
Web Services for Remote Portlets

Dave Landers
WebLogic Portal Architect
BEA Systems, Inc.
Session Goals

- Basic WSRP description
  - Outline of protocol
  - Why / when WSRP is useful
- Developer best practices
  - Deploy your portlets locally or with WSRP
  - Information to avoid problems
- Non-Goals
  - Implementing a WSRP-enabled Portal
  - Reading WSRP SOAP messages
Overview
What Is WSRP?

- An OASIS standard
  - Version 1.0: August, 2003
  - Version 2.0: In the works....

- Aligned with other portlet specifications
  - JSR-168: Java Portlet Specification
  - .NET
  - etc.
Web Services for Remote Portlets

- **Web Service**
  - A Protocol for communications
  - A Contract for behavior

- **Portlets**
  - User Interfaces aggregated in a portal
  - Mini-applications, displays, widgets, *etc.*

- **Remote**
  - Portlets are hosted separately from the Portal
What Can WSRP Do?

- Deliver portlets to multiple portals
- Aggregate portlets from several providers
- Provide a UI-oriented service
  - Rather than data- or logic-based services
- Unify Portlet standards
  - Ensure concepts and data exchanged are aligned with other standards in both the portal and web service arenas.

WSRP Overview - Goals
Why WSRP?

- **Aggregate**
  - Portlets from several sources
    - And maybe deployed on different platforms
- **Centralize**
  - Access for your users
  - Unify several individual portals into one “master portal”
    - Unify your intra-net
- **Decouple**
  - Portals from portlets; Portlets from each other
  - Helps with: Deployment, Administration, Development, Upgrade, *etc.*
WSRP Support

- Apache WSRP4J
- BEA WebLogic Portal
- BEA AquaLogic User Interaction (Plumtree)
- Clickmarks
- eXo (open source)
- Fujitsu
- Gluecode
- IBM WebSphere Portal
- Intrafinity

- Liferay (open source)
- Microsoft SharePoint
- NetUnity
- OracleAS Portal
- SAP
- Sun
- uPortal (open source)
- Vignette
- webMethods

and more ...
A UI Oriented Service

- Much Higher level than other services

- Compare and Contrast:
  - Data or Logic services
    - Retrieve data
    - Interact with business logic functions
    - Each user builds another UI
  - WSRP
    - Retrieve the UI markup
    - Interact with the UI
    - Hides the details of data or logic, focuses on the UI
WSRP Basics
Basic WSRP Operations

- Get Markup
  - HTML fragments

- Handle interactions
  - Forms, links, *etc.*

- Service Description
  - Producer advertises its capabilities and requirements

- Registration
  - Consumer registers with Producer

- Customization
  - Of portlets
Producer and Consumer

**Producer**
- The Web Service
- Offers one or more Portlets
- Not necessarily a Portal itself

**Consumer**
- The Web Service Client
- Offers Portlets from one or more Producers
- Is usually a Portal
- Mediates interaction between User and Producer
WSRP Interfaces

- WSRP defines 4 interfaces (WSDL)
  - Service Description *
  - Registration
  - Markup *
  - Portlet Management
    - Only two are required (*)
    - Not all operations of these are required
- Different support requirements for Producers and Consumers
  - Several levels of functionality / complexity
Service Description Interface

- Required
- Producer provides its description
  - Capabilities
  - Requirements
    - Is registration required?
    - Require cookie initialization
  - Portlet offerings
Registration Interface

- Optional
  - Producers are not required to implement it
  - Producer may require it of Consumers

- Consumers register with Producer
  - May include “out-of-band” communications
    - Phone calls or email or payment...
    - ... to get registration keys, etc.

- Registration can be used by Producer to
  - Provide enhanced offerings to some Consumers
  - Provide Consumer-specific offerings
  - Customize portlets per-Consumer
Markup Interface

- Required
  - The Main Thing
- Get Markup
  - The display or UI to render
- Perform Interaction
  - As in, post a form
- Some Session and Cookie stuff
Portlet Management Interface

- Optional
- Allows Consumers to manage Portlets
- Portlet descriptions
- Persistent state
  - Personalization
  - Get / Set / Describe properties
- Portlet lifecycle
  - Cloning and destroying portlet instances
Two-Step Protocol

- WSRP uses a “Two Step” protocol for interaction
  - `performBlockingInteraction`
    - Like a HTTP POST
    - Returns new portlet state, *etc.*
    - JSR-168: `processAction()`
  - `getMarkup`
    - Using any state returned by Producer
    - Returns markup to display
    - Repeated calls return same markup
    - JSR-168: `render()`
Doing the Two Step

- Allows Consumer to get markup as needed
  - Consumer may repeatedly render the portlet
    - While user interacts with other portlets
    - This requires separation between interaction and rendering

- Producer may return markup with performBlockingInteraction response
  - Optional optimization
Caching of Markup

- Consumer may cache markup
  - Avoid repeated calls to getMarkup
  - Producer informs Consumer if caching is OK
    - Returns CacheControl element with markup
      - Expiration of markup validity
      - User scope (shared or not)

- Producers may also cache
  - As appropriate
Portlet State

- **Transient state**
  - Navigational state
    - “Where are we” in a portlet navigation cycle
    - Bookmarkable
  - Session state
    - Similar to HTTP Session data

- **Persistent state**
  - Properties exposed to Consumer
  - Some other state (database, etc.)
State Management

- Consumer and Producer cooperate to maintain Portlet State
- Producer returns to Consumer
  - Navigational State
  - Session ID
  - Handle of cloned portlets
- Consumers return these with future requests
Personalization

- Portlets can be personalized
  - Properties (persistent state) stored per-user
  - User might have several instances of the same portlet, each customized differently
  - JSR-168: PortletPreferences

- Requires new portlet instance
  - Unique portlet handle identifies instance
    - Consumer-Configured portlet
  - Portlets cloned to make instances
Portlet Cloning

- Portlet handle returned by Service Description is generic or shared
  - Producer-Offered portlet
  - Not customized

- When changing persistent state, Producer must:
  - Clone portlet
  - Return new portlet handle
  - Consumer uses this handle from now on
Cloning Mechanisms

- **Explicit**
  - Portlet Management Interface: `clonePortlet`

- **Implicit**
  - Markup Interface: `performBlockingAction` with `cloneBeforeWrite`

```
Producer
Offered
(Not Modifiable)

Consumer
Configured
(Modifiable)

setPortletProperties

clonePortlet

destroyPortlet

performBlockingAction

cloneBeforeWrite
```
WSRP Basics - Summary

- Interfaces
  - Many optional levels of support
- Markup and Interaction
  - Two Step Protocol
- State
  - Transient and Persistent
- Cloning Portlets
  - So they can be personalized
Writing for WSRP

Best Practices
Writing Portlets for WSRP

- If you write for WSRP, your portlets will still work as local portlets
- You *can* write local portlets that will break with WSRP

Biggest issues are
  - Rewriting URLs
  - Name-spacing of identifiers
  - Assuming things about the container
    - Availability of HTTP Request and Session data
  - Assuming relationships between portlets
    - Where portlets are deployed (relative to each other)
    - Using Request or Session data for communications
## Markup Considerations

- **Portlet returns Markup Fragment**
  - Usually assembled into a full page (portal)
    - By Consumer’s Portal
  - Just like local portlets

- **Implications**
  - Markup Fragment is not a full document (page)
  - Identifiers must co-exist with remainder of page
    - Named attributes
    - Javascript functions, variables, etc.
Markup as Fragments

- Some tags are not allowed
  - `<html>`
  - `<body>`
  - `<head>`
  - `<title>`
  - `<frame>`
  - `<frameset>`

- Consumer will assemble fragments into larger portal page
Names and Identifiers

- HTML forms, Javascript methods, variables
  - Several portlets are aggregated on a portal page
  - Many names need to be unique to a portlet
    - Avoid collisions between Portlets
    - Even two instances of same portlet
Namespace Encoding

- Consumer Rewriting
  - Portlet prefixes names with “wsrp_rewrite_”
  - Consumer replaces this with something unique
    - And valid for Javascript variables, etc.

- Producer Encoding
  - Portlet uses namespacePrefix provided by the Consumer to prefix tokens in the markup

- Best Practice for Portlet Developers
  - Use available APIs and Tags
    - JSR-168: renderResponse.getNamespace() + “foo”
URLs in Remote Portlets

- URLs used to render portlets or perform actions (POST)
  - Can not be “normal” URLs
  - Refer to Consumer’s portal
  - May need to “target” a specific portlet
  - Need to forward to Producer *via* WSRP

- No way for portlet developer to know what the URL should look like
Consumer URL Rewriting

- Required functionality by WSRP
- Portlet writes URLs using tokens
  - Consumer replaces tokens to make URLs
    - `wsrp_rewrite?wsrp-urlType=render&wsrp-mode=help&wsrp-windowState=maximized/wsrp_rewrite`
- URLs provided by Producer contain tokens for
  - Type of URL (render, resource, action, etc.)
  - Portlet State
  - Mode, Window State
  - ... etc.
  - And no specific host names, etc.
Producer URL Rewriting

- Optional in WSRP
- Consumer supplies templates to Producer for various types of URLs
  
  \[ http://consumer.com/path/{wsrp-urlType} \]
  \[ ?mode={wsrp-mode}&var=name& \ldots \]

- Producer rewrites URLs, replacing \{values\} in template

- Probably more efficient than consumer rewriting
URLs in Portlets

- Don’t hard-code them
  - Except for absolute off-site URLs (links)
- Anything pointing to the Portal or Portlet must be rewritten
  - By Consumer or Producer

- Best Practice for Portlet developers
  - Use available APIs and Tags
    - Let the Producer / Consumer figure it out
    - JSR-168: `createActionURL()` and `createRenderURL()`
Dynamic Client-Side Code

- Dynamically generated URLs and Identifiers have problems
  - Those computed by client-side (Javascript) code
  - Javascript generated by Producer must understand consumer templates
  - Store URL templates and namespace in the markup (script)
    - Compute URLs and names using these
Request and Session Data

- May not be accessible by all WSRP portlets
  - Portlets may be remote from each other
  - May not be accessed by same Request
    - Is a Web Service request (may not see HTTP)
    - HTTP Request “belongs to” the Portal
  - Don’t use Request attributes or Session data as inter-portlet communication scheme
    - If you must... Session-sharing portlets should be “grouped” on same Producer
      - Ensure your Producer supports this
Inter-Portlet Communications

- Sometimes, portlets must interact
  - Two views of same data
  - Send events to each other
  - ... or otherwise cooperate
- Can be difficult to arrange
  - Timing is especially difficult
    - Rendering of portlets is not sequential or synchronous
- Remote adds an extra complication
Inter-Portlet Relationships

- May need specific deployment requirements
  - All related portlets on same Producer
  - All portlets run in one JVM? (clustering issues)
- Some vendors may have useful features
  - WebLogic Portal has IPC Events
    - And associated WSRP Extension
- WSRP 2.0 is addressing portlet events
  - I suggest designing any inter-portlet communications as events
CSS Style Sheets

- WSRP specifies definitions for CSS
  - Portlets from several sources can have a common look-and-feel
    - Local portlets
    - Several Producers
  - Cover a lot of types
    - Fonts, Tables, Forms, Messages, Links, Menus, etc.
    - See WSRP Specification for full list (Section 10)

- Portlet container should offer same styles...
Optional WSRP Features

- Consumers must support more than Producers
  - But there are still a lot of options

- Portlet Developers
  - Know features of your Producer
    - Cloning (personalization) support
    - Portlet persistent state
    - Modes and window states

- Portal Administrators
  - Understand your Consumer features
  - Choose a Consumer that is capable “enough”
    - To handle Producers you expect
WSRP and Security

- Several concerns
  - Registration
  - Authentication
  - Authorization
  - Privacy
  - Integrity

- Same issues as other Web Services
  - Mostly handled outside WSRP
Web Services Security

- Document level security
  - WS-Security
  - SAML
  - XML-Signature
  - XML-Encryption

- Transport level security
  - HTTPS

- Consumer Authentication
  - SSL with Client certificates
WSRP Security

- Consumer Authorization
  - Registration Interface
  - Including any necessary out-of-band process

- End-User Access Control
  - Consumer-side
    - Portal login, entitlements, etc.
  - Producer-side
    - Using userContextKey in Markup Request
      - Insecure, not authenticated - real purpose is personalization

Consumer / Producer may use Single Sign-On
Error Handling Suggestions

- Handle errors in your portlet
  - Gracefully
    - Suitable business message
  - Rather than propagating exceptions
    - Exception --> Producer error --> WSRP Fault --> Consumer error --> User has no idea what happened

- Use meaningful error pages on Consumer side
  - Rather than relying on “StackTrace (or WSRP Fault) Inside Portlet” pattern
Debugging

- Two usual problems
  - The portlet itself
    - Debug as local portlet \((i.e.\) in Producer’s Portal)\)
  - Problems when portlet deployed with WSRP
    - Lots of layers to unravel
    - Snoop on SOAP messages
    - Producer faults described in WSRP Primer
  - Remember issues with
    - URL Rewriting
    - Identifier Namespacing
    - Request and Session reliance
    - Consumer / Producer complexity level mismatch
Summary

- Decoupling Portlets can simplify Deployment, Administration, Interoperability, Development Lifecycle, etc.
- Aggregate Portlets from several sources
- Writing for WSRP will not break local usage
  - Gives you options
- WSRP attempts to unify concepts in other Portlet specifications
  - JSR-168, .NET, etc.
References

- Oasis
    - Specification
    - White Papers
    - Primer
Other Related Sessions

- Noel Bergman
  - Introduction to Portlet Programming with JSR-168

- Kelvin Lawrence
  - Web Services Advanced Topics
    - Security
The End

Please fill out the evaluations

dave.landers@bea.com