JBoss World 2006 LAS VEGAS

WSRP Support in JBoss Portal

Christophe Laprun
Julien Viet
JBoss, a division of Red Hat

- Christophe Laprun
  - ✓ JBoss Portal developer
  - ✓ WSRP lead
- Julien Viet
  - ✓ JBoss Portal founder and project lead
  - ✓ JBoss Inc. representative on the JSR 286 Portlet 2.0 Expert Group

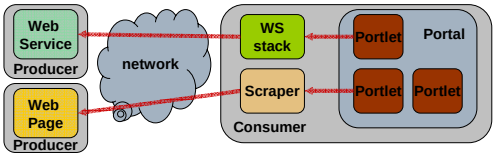
Agenda

- Introduction to WSRP
- Architectural overview
- Roadmap

Motivation

- Portals offer aggregation of content from diverse sources in the form of portlets
- Content can be:
  - ✓ Local to the portal server
  - ✓ Remote via web services or content scrapping
- Local content is addressed by JSR 168
- Remote content is addressed by WSRP

Problems with remote content



- No defined contract between producers and consumers
- Producer-specific development on consumers
- Difficult for producers to evolve/deploy content

Enters WSRP

- Content scraping is not a viable solution
- Web services are **data**-oriented but portals are **presentation**-oriented

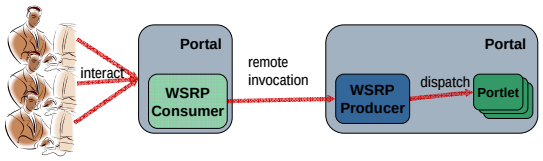
⇒ WSRP was defined to offer a standard solution to present remote content in portals

What is WSRP?

- Web Services for Remote Portlets
- Provides a standard for interactive, presentation-oriented web services
- Approved as an OASIS standard in August 2003
- WSRP 2.0 is being worked on
  - ✓ Public review just started

WSRP overview

- Two actors: Consumer and Producer
  - ✓ Producer publishes content via WSRP interface
  - ✓ Consumer interacts with WSRP services on behalf of users



Benefits

- WSRP clearly defines the contract between producers and consumers
- Provides interoperability between middleware stacks
- Easier development on the consumer side:
  - ✓ All WSRP content is presentation-oriented
  - ✓ Access is done in a standard way
- Easier management on the producer side:
  - ✓ Retention of content control
  - ✓ Subscription management
- Built on existing standards: SOAP, WSDL…

WSRP specification

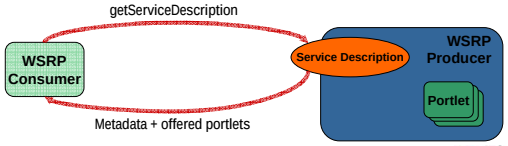
- Provides a WSDL web service definition for invocation of WSRP services
- Defines semantics for WSRP interactions between Consumers and Producers
- Defines rules for markup generation (producer side) and aggregation (consumer side)
- Implementation language agnostic

WSRP interfaces overview

- Service Description interface (required):
  - ✓ acquiring the Producer's and portlets metadata
- Markup interface (required):
  - ✓ retrieving portlets markup and processing user interaction
  - ✓ Consumer assistance to producer operations
- Registration interface (optional):
  - ✓ establishing, updating and destroying a registration
- Portlet Management interface (optional):
  - ✓ getting Portlet metadata
  - ✓ cloning Portlets for further customization
  - ✓ interacting with portlet properties

Service description interface (required)

- Provides producer metadata
  - ✓ Requires registration?
  - ✓ Requires cookie support?
  - ✓ Custom modes and window states
  - ✓ Supported locales
  - ✓ …
- Provides offered portlets description:
  - ✓ Handle
  - ✓ Supported markup types
  - ✓ Title, description
  - ✓ …



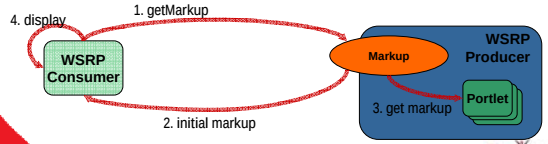
Markup interface (required)

- Retrieve markup content for a given portlet
- Notify a portlet of user interaction
- Support producer initialization of cookies
- Release session-related resources

13

Markup interface (cont.)

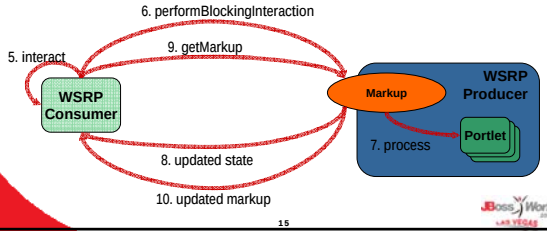
- Markup is aggregated by the consumer:
  - ✓ Portlets generate markup fragments
  - ✓ URLs should refer to the Consumer, not the Producer:
    - Consumer URL rewriting
    - Producer URL writing
- Producer/portlets can return state that the Consumer is required to provide back



14

Markup interface (cont.)

- Two-step protocol similar to JSR-168's:
  - ✓ Blocking processing of user interaction and state updating
  - ✓ Markup rendering (which shouldn't impact the portlet state)



15

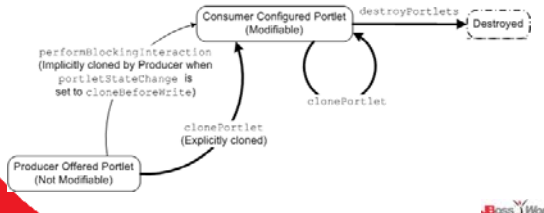
Registration interface (optional)

- Register with the Producer:
  - ✓ Can associate portlet and customization data with consumer
  - ✓ Can track portlet usage
  - ✓ Can restrict/tailor portlets access
  - ✓ ...
- Update registration data
- Deregister
- Two kinds of registration:
  - ✓ In-band: registration is completely handled via the registration interface
  - ✓ Out-of-band: registration requires additional producer/consumer interaction that is not covered by WSRP

16

Portlet management interface (optional)

- Retrieve portlet metadata
- Retrieve portlet properties metadata
- Interact with portlet properties
- Manage portlet lifecycle



17

Goals for JBoss' implementation

- Provide a superior open source WSRP implementation
  - ✓ Improved manageability and reusability
  - ✓ Performant, reliable, configurable
- Separation of concerns
  - ✓ WSRP is just another modality to interact with portlet content
  - ✓ WSRP operations are transparently handled
- API/component model agnostic to allow for easy exposition of legacy content to WSRP

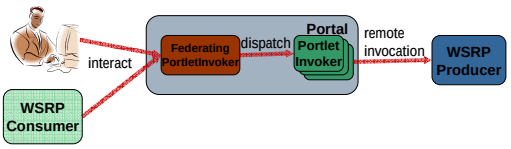
18

PortletInvoker concept

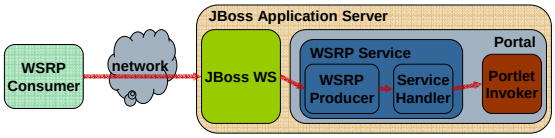
- PortletInvoker presents a unified view of portlet operations
- All portlet invocations go through a PortletInvoker

PortletInvoker concept (cont.)

- FederatingPortletInvoker:
  - ✓ Federates heterogeneous invokers
  - ✓ Routes the invocation to the appropriate invoker
- WSRP is just another modality



Producer implementation

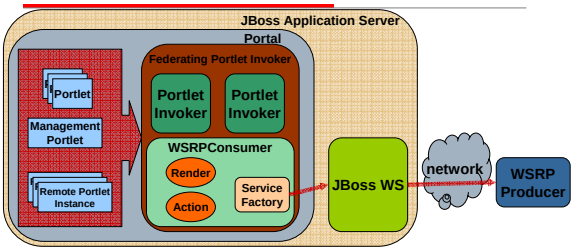


1. Consumer sends WSRP request (via SOAP)
2. JBoss WS dispatches the SOAP message to the WSRP service endpoint: WSRPProducer
3. Producer dispatches the request to the appropriate handler
4. Handler examines the WSRP request, extracts the information and creates a PortletInvocation object that is then processed by a PortletInvoker
5. Invocation result is examined and a WSRP response is created by the handler and propagated back up the chain to the Consumer.

Traditional approach to consumers

- So-called *Proxy Portlet* solution
- Each remote portlet maps to a local portlet acting as a proxy for the remote producer
- Need to deploy/clone a portlet per remotely offered portlet
  - ✓ Burden on portal administrator
  - ✓ Problematic for user customization

JBoss' approach



- WSRPConsumer is a PortletInvoker!
- Consumer is configured to access a specific producer
  - ✓ Instances can be connected to remote portlets just as easily as to regular portlets
- Consumer decides which WSRP request to issue based on the called PortletInvoker method

Benefits

- Easier administration
- True middleware approach
- WSRP is transparent
  - ✓ Local and remote portlets are handled the same way
- Reusability:
  - ✓ Portal can present JSR-168 or WSRP portlets in the same page
  - ✓ Possible to create a PortletInvoker fronting legacy content and expose it via WSRP

WSRP implementation in 2.4

- Producer:
  - ✓ Base level support
    - Service Description and Markup interfaces
- Consumer:
  - ✓ Base level support
    - Service Description and Markup interfaces
  - ✓ Support for standard window states and modes
  - ✓ Support for simple registration (String properties)
  - ✓ Basic caching

25

Integration in JBoss Portal

- Runs only in JBoss Portal
- Packaged as portal-wsrp.sar
  - ✓ SOAP endpoints
  - ✓ Producer services and stack
  - ✓ Consumer services and stack
- Consumer configuration via **`*-wsrp.xml`** files
  - ✓ Provide information on remote producers
  - ✓ As easy as dropping a descriptor in **`/deploy`**

26

Consumer configuration example

- The producer is uniquely referenced as 'someid'
- The producer metadata are considered valid for 2 minutes
- End points can also be configured with the WSDL url

```
<deployments>
<deployment>
<wsrp-producer>
  <producer-id>someid</producer-id>
  <expiration-cache>120</expiration-cache>
  <endpoint-config>
    <service-description-url>…</service-description-url>
    <markup-url>…</markup-url>
    <registration-url>…</registration-url>
    <portlet-management-url>…</portlet-management-url>
  </endpoint-config>
</wsrp-producer>
</deployment>
</deployments>
```

27

Producer configuration example

- **`jboss-portal.sar/portal-wsrp.sar`** is the service that adds WSRP remoteness
- Publishing a portlet via WSRP is done by defining the portlet as remotable in **`jboss-portlet.xml`**

```
<portlet-app>
   <portlet>
      <portlet-name>MyPortlet</portlet-name>
      <remotable>true</remotable>
   </portlet>
</portlet-app>
```

28

WSRP 2.0

- Public draft review just begun
- Customer mediated coordination
  - ✓ Events: interaction is now a three-step protocol (**`handleEvents`**)
  - ✓ States
- Better portlet management
  - ✓ State migration: import/export, copy
  - ✓ Lifetime management: automatic cleanup of portlet resources
- Parameters passing made more explicit
  - ✓ Provide navigation coordination

29

WSRP Roadmap

- WSRP for JBoss Portal 2.6
  - ✓ Portlet management interface
  - ✓ Registration interface
  - ✓ Management GUI
  - ✓ Bonus features
    - Bypass SOAP layer and use JBoss http invoker + JBoss Serialization
      - ✓ Same semantic : remote portlet invocation
      - ✓ Up to 10 time faster !!!
    - Transparent security propagation
- WSRP for JBoss Portal 3.0
  - ✓ WSRP 2.0

30

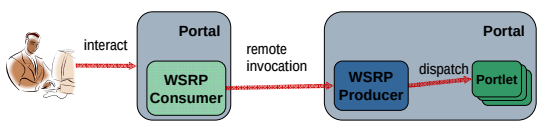
References

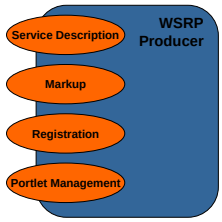
- JBoss Portal:
  - ✓ http://www.jboss.com/products/portal
- WSRP TC web site:
  - ✓ http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp

Other relevant sessions

- Portlet 2.0 preview : tomorrow 9am
- Don't miss it!

Q & A





Service description interface (required)

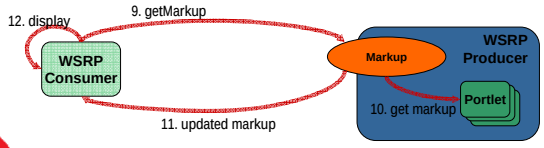
- Provides producer metadata:
  - ✓ Requires registration?
  - ✓ Requires cookie support?
  - ✓ Custom modes and window states
  - ✓ Supported locales
  - ✓ ...
- Description of offered portlets and properties:
  - ✓ Handle
  - ✓ Supported markup types
  - ✓ Title, description
  - ✓ ...

Markup details

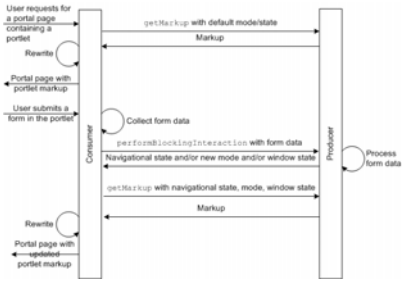
- Two-step protocol similar to JSR-168's:
  - ✓ Blocking processing of user interaction and state updating
  - ✓ Markup rendering (which shouldn't impact the portlet state)
- Portlet markup is aggregated by the consumer:
  - ✓ Portlets generate markup fragments
  - ✓ URLs should refer to the Consumer, not the Producer:
    - Consumer URL rewriting
    - Producer URL writing
- Producer/portlets can return state that the Consumer is required to provide back

Markup Interface (end)

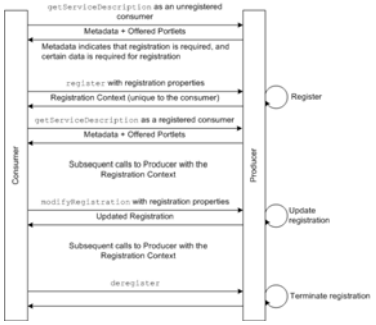
- Retrieve markup content for a given portlet
- Notify a portlet of user interaction
- Support producer initialization of cookies
- Release session-related resources



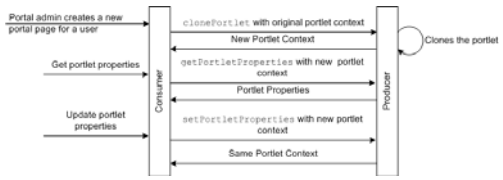
Typical markup interaction



Typical registration sequence



Typical portlet cloning interaction



Proposed roadmap for 2.6 (Sept. '06)

- Producer:
  - ✓ Simple level support (full Portlet Management interface support)
  - ✓ Caching support
  - ✓ Clustering support
  - ✓ URL templating support
  - ✓ Persistent local state
  - ✓ Parallel rendering support
- Consumer:
  - ✓ Medium level support (full Portlet Management Interface support)
  - ✓ Explicit property setting mechanism with GUI for property management
  - ✓ Parallel rendering support
  - ✓ Localization

43

WSRP Profiles

- Conformance levels defined to help organize conformance testing
- Different levels for Producer and Consumer

44

Producer conformance levels

- Base
  - ✓ Implements only the MUST interfaces
  - ✓ No state (session or persistent); opaque state sent back to Consumer
  - ✓ No cloning
  - ✓ No initialization required
  - ✓ Does not rewrite URLs in markup
  - ✓ Does not require registration
- Simple
  - ✓ May request initialization; could store state in cookies
  - ✓ Supports cloning
  - ✓ May require registration (out-of-band).
  - ✓ Session state; creates and sends session handles to the Consumer
- Complex:
  - ✓ May rewrite URLs (requires Consumer templates)
  - ✓ May offer both in-band and out-of- band registration
  - ✓ Persistent local state
  - ✓ May support grouping of portlets

45

Consumer conformance levels

- Base
  - ✓ Implements only the MUST interfaces
  - ✓ VIEW mode, NORMAL window state only
  - ✓ Supplies no user information
  - ✓ Rewrites URLs
  - ✓ Initializes the Producer if required
  - ✓ Handles Producer cookies
  - ✓ Limited markup types (e.g. html)
  - ✓ Does not clone
  - ✓ No in-band registration
- Simple
  - ✓ Support for standard modes and window states
  - ✓ Support for in-band registration
  - ✓ Supplies basic user information
  - ✓ Handles implicit clones
- Medium
  - ✓ Complex user management
  - ✓ Multiple markup types
  - ✓ Caching according to Producer-supplied cache control
  - ✓ May explicitly clone portlets
  - ✓ May supply URL rewrite templates to Producers supporting URL rewriting
- Complex
  - ✓ May support custom window states and/or modes.
  - ✓ Multiple levels of user access
  - ✓ Localization
  - ✓ May use explicit property-setting mechanism; create custom UI for property management.

46