Federated Portal Network – an Inside Look: Part II

Applies to:

SAP Enterprise Portal 7.0

Summary

This is a continuation from the previous article Federated Portal Network – an inside Look: Part I, drilling further into analyzing the essence of Federated Portal Network Architecture, with reference to WSRP.

Author: Milton Ghosh

Company: HCL Technologies

Created on: 20 July 2007

Author Bio

Milton Ghosh is working as a NetWeaver consultant for HCL Technologies.

Table of Contents

A Brief Recall of Federated Portal Network

As we had seen Federated Portal was a breakthrough in the world of Portals providing an abstract Distributed Portal environment to the end users without the need of overloading a single Portal with contents & resources. It achieved seamlessly the requirement of sharing contents between its component portals and decoupling their resources thus reducing maintenance cost considerably.

Web Service for Remote Portlets

The content sources (portlets, iViews whatever one may like to call them) that are hosted on a remote server needs a standard protocol to be accessed and displayed by content aggregators vis-à-vis Portals. Web Services for Remote Portlets (WSRP) is the protocol designed for the same allowing a local portal to consume portlets hosted by a remote producer. WSRP is nothing more than an XML based Web Services specification that will allow for the plug-n-play of portals.

Need for WSRP: WSRP, JSR-168 and Web Services.

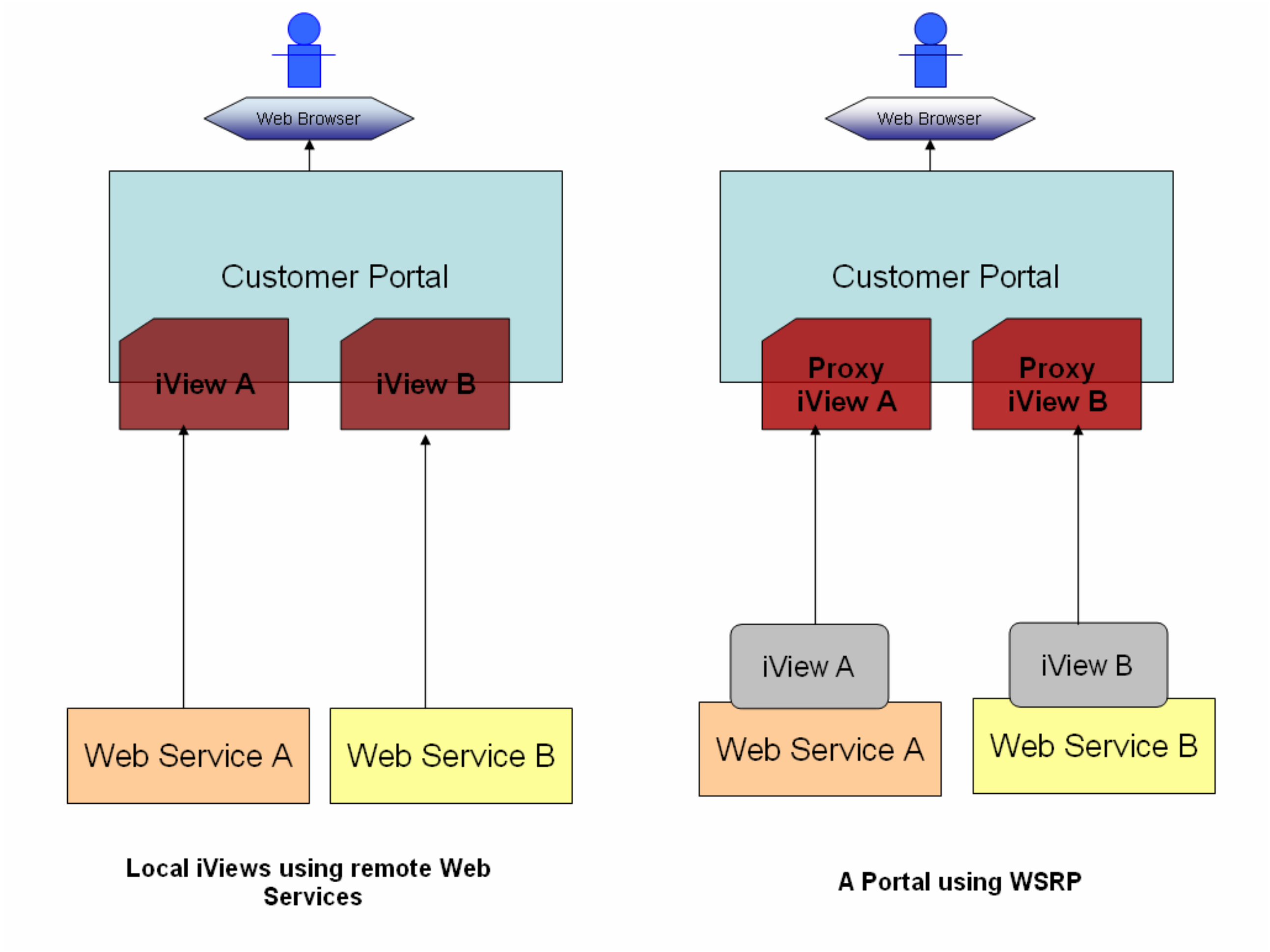
JSR-168 specification defines a set of APIs for portlets and addresses standardization for preferences, user information, portlet requests and responses, deployment packaging, and security. Adhering to this standard the developed portlets becomes portal server independent making them executable over most portal servers. The W3C defines a **Web service** (many sources also capitalize the second word, as in Web Services) as a software system designed to support interoperable Machine to Machine interaction over a network. Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. So the question arises while we have Web Services on one hand offering mechanism to create platform-independent services and JSR – 168 on the other hand defining a structured standard to develop portlets, was **WSRP** a necessary? The answer lies in the fact that while Web services give you the ability to reuse back-end services, WSRP lets you reuse the entire user interface.

WSRP provides an interface to deal with issues such as Localization, Mime types, portlet modes (for example, edit, help and view modes), and window states (for example minimized, maximized, solo, and normal window states). Defining these states for each web service would be tedious and error-prone.

For instance, one might want to provide the end-users with a portal having the ability to look-up the price of a stock by entering a ticker (Character printer that automatically prints stock quotations on tickertape) symbol. Web service which provides precisely this capability is abundant and upon encountering such a service, one would next need to code a client to bind to and consume the Web service, as well as develop a portlet to allow portal users to interact with the service. Now while Web Service makes life easy for developing the Web Service client developing the user interface is a heavy duty task along with the need to go through the process of deploying the newly developed portlet and Web services client on your portal server. The process of developing and deploying the front-end application needed to make the functionality usable by a portal user is tedious and time-consuming. WSRP comes as the savior, by providing presentation-oriented, interactive web services, instead of just providing raw data or single business functions that still require special rendering on the portal side.

The basic difference between WSRP and JSR-168 can be summed up in the following statement "*WSRP is a standard messaging interface for interacting with compliant UI components. JSR-168 is a standard Java interface for portlets that builds on the J2EE programming model of servlets. This is an interface between a particular Java type of UI component and its hosting container. Other differences are that JSR-168 is Java only, while WSRP is platform independent. JSR-168 generally is local while WSRP is remote*"

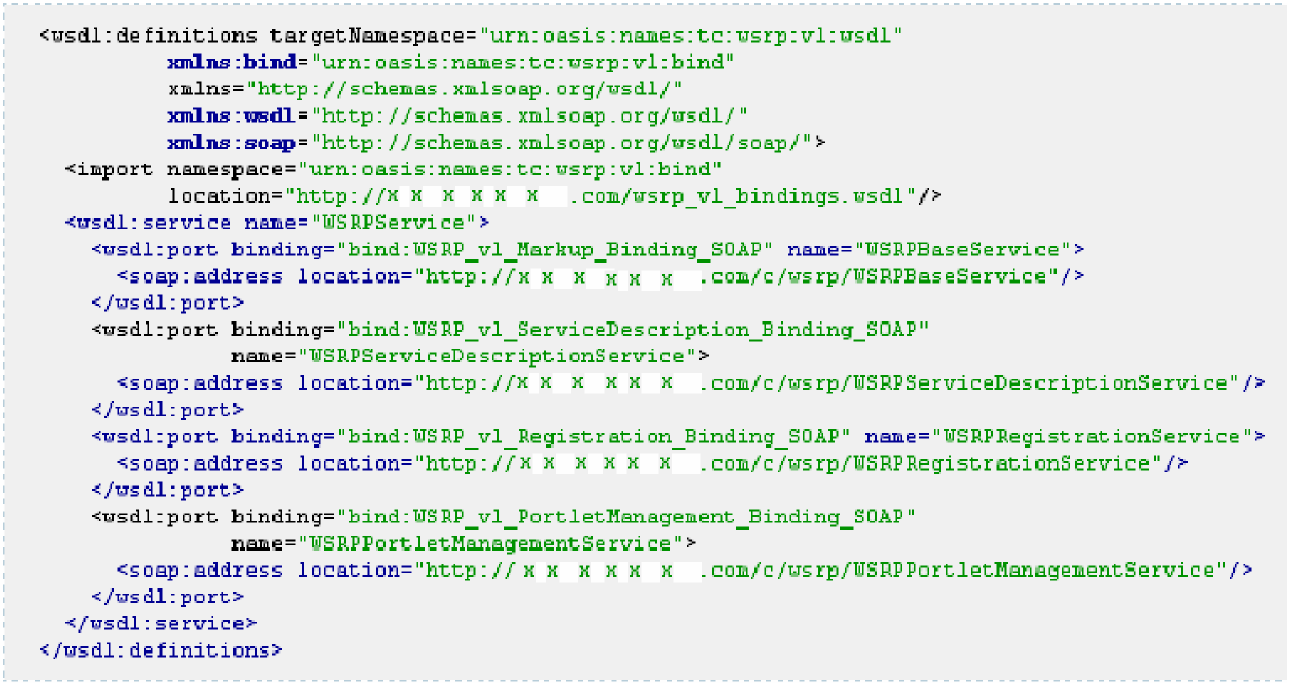
A comparative diagram between normal web service based portlets and WSRP consumed portlets



It is much more convenient if the A and B web services would include application *and presentation* logic and directly produce markup fragments that are easy to aggregate at the consuming portal. Moreover the markup fragments would be reusable components. WSRP does just that.

Closer Look at WSRP

Let's have a closer look at a sample WSDL for WSRP. This WSDL contains the necessary information the consumer requires to communicate with the remote producer.



```
<wsdl:definitions targetNamespace="urn:oasis:names:tc:wsrp:v1:wsdl"
          xmlns:bind="urn:oasis:names:tc:wsrp:v1:bind"
          xmlns="http://schemas.xmlsoap.org/wsdl/"
          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <import namespace="urn:oasis:names:tc:wsrp:v1:bind"
          location="http://x x  x x x  x   .com/wsrp_v1_bindings.wsdl"/>
  <wsdl:service name="WSRPService">
    <wsdl:port binding="bind:WSRP_v1_Markup_Binding_SOAP" name="WSRPBaseService">
      <soap:address location="http://x x  x  x x  x   .com/c/wsrp/WSRPBaseService"/>
    </wsdl:port>
    <wsdl:port binding="bind:WSRP_v1_ServiceDescription_Binding_SOAP"
               name="WSRPServiceDescriptionService">
      <soap:address location="http://x x  x  x x  x   .com/c/wsrp/WSRPServiceDescriptionService"/>
    </wsdl:port>
    <wsdl:port binding="bind:WSRP_v1_Registration_Binding_SOAP" name="WSRPRegistrationService">
      <soap:address location="http://x x  x  x x  x   .com/c/wsrp/WSRPRegistrationService"/>
    </wsdl:port>
    <wsdl:port binding="bind:WSRP_v1_PortletManagement_Binding_SOAP"
               name="WSRPPortletManagementService">
      <soap:address location="http:// x x  x  x x  x   .com/c/wsrp/WSRPPortletManagementService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

We can see clearly the WSRP implemented interfaces (defined by port types) can be categorized into four out of which first two are mandatory and the later two being optional.

- **WSRPBaseService (Mark-up Interface):** This Interface allows remote interaction between WSRP consumer with a WSRP producer portlet.
- **WSRPServiceDescriptionService (Service Description Interface):** This Interface allows a WSRP producer to promote its capabilities to perspective consumers. A WSRP consumer may invoke the getServiceDescription operation of the interface to query about the portlets offered by a remote producer, as well as additional metadata about the producer itself.
- **WSRPRegistrationService (Registration Interface):** This optional Interface allows a WSRP producer to require that WSRP consumers perform some sort of registration before they can interact with the service through the other two required interfaces. Through this mechanism a producer can customize its relation with a specific type of consumer.
- **WSRPPortletManagementService (Portlet Management Interface):** The Portlet Management Interface gives the WSRP consumer access to the life cycle of the remotely-running portlet

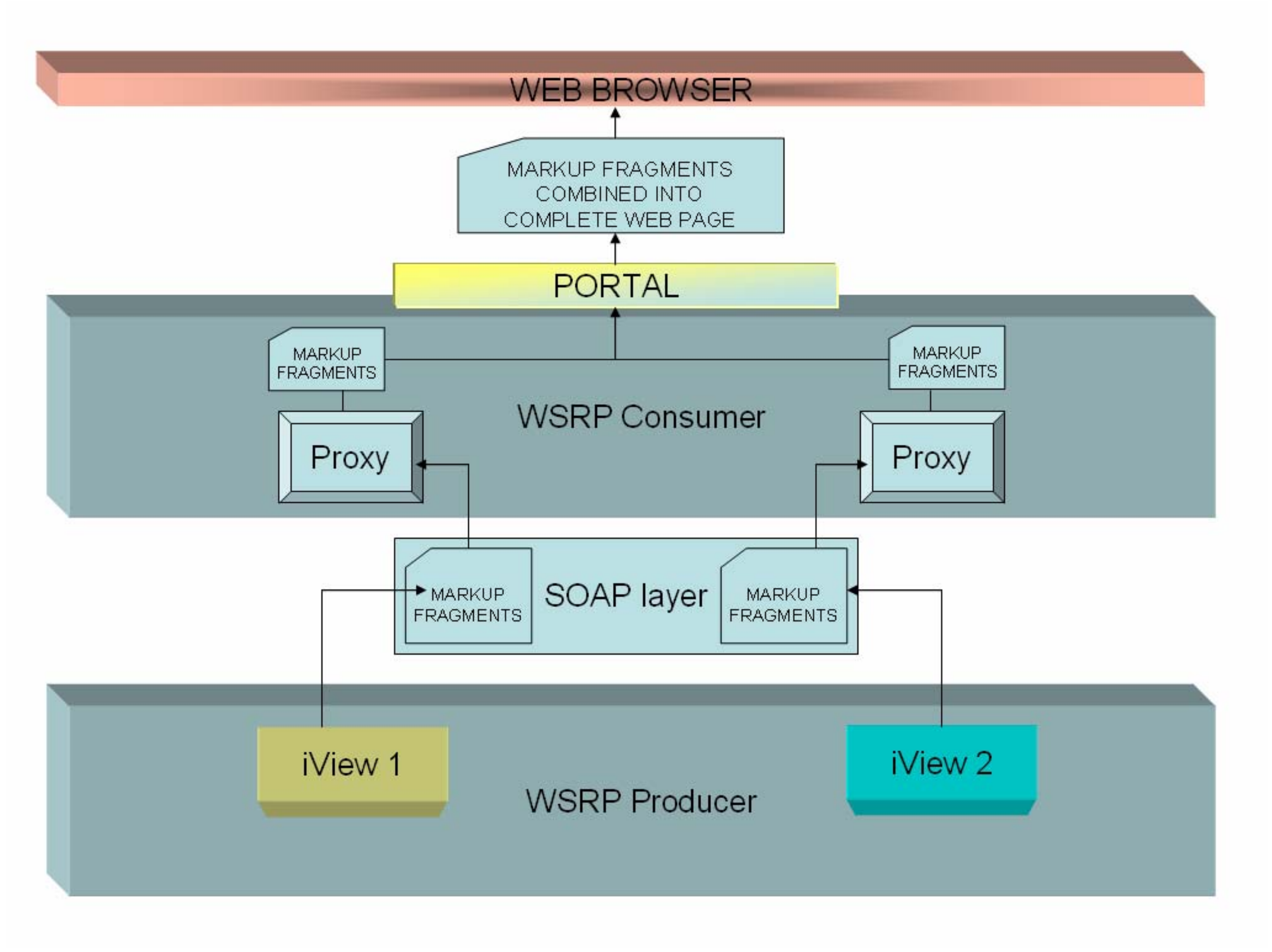
WSRP – In Action

So long we have been referring about producers and consumers. Now let's have a formal definition of both of them to have a better understanding of the same.

The Producer is a web service that offers one or more portlets and implements various WSRP interfaces/operations. Depending on the implementation, a producer may offer just one portlet, or may provide a runtime (or a container) for deploying and managing several portlets.

The Consumer is a web service client (typically a portal) that invokes producer-offered WSRP web services and provides an environment for users to interact with portlets offered by one or more such Producers.

The following diagram shows how a WSRP Consumer Portal aggregates mark-up from remote portlets and displays them.



SAP Enterprise Portal includes a WSRP Consumer Framework available from the Federated Portal link under the System Administration tab. To surface a remote portlet, SAP provides the Proxy-to-Portlet iView.

The advent of WSRP means that the developers are no longer limited to being able to reuse only back-end services, but can instead reuse presentation-oriented services as well.

We will continue our journey further into Federated Portal Network in my forthcoming articles, looking deeper into WSRP and their relevance to SAP Enterprise Portal.

Related Content

Please include at least three references to SDN documents or web pages.

- Implementing a Federated Portal Network
  https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/3eea14b9-0a01-0010-06b9-8410ab7675f6
- http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp
- WSRP 1.0 Frequently Asked Questions
  http://www.oasis-open.org/committees/download.php/11774/wsrp-faq-draft-0.30.html
- Web Services for Remote Portlets 1.0 Primer
  http://www.oasis-open.org/committees/download.php/21178/wsrp-primer-1.0.html

Disclaimer and Liability Notice