Presentation Oriented Web Services

Jana Polgar
Monash University, Australia

VISION FOR USER-FACING PORTLETS

Web services introduced the means for integrating and sharing business processes via the Internet. WSRP's (WSRP specification version 1, 2003) goal is to extend the integration further by providing a framework for sharing Web service presentation components. WSRP specification formulated a standard protocol, which enables all content and application providers to create Web services, generate their presentation faces as HTML fragments, and offer them to the consumers to be plugged into their local portals.

Portals and portlets (JSR 168, 2005) provide specific presentation logic to aggregate data from multiple sources, which could be legacy systems, Enterprise Information Systems (EIS), local or remote Web services, or EIS with exposed Web service interfaces.

The WSRP specification is intended for presentation-oriented Web services, and user-facing Web services that can be easily integrated with portals. They let businesses provide content or applications without requiring any manual content or application-specific adaptation by portal presentation logic. It is envisaged that in the near future portals will easily aggregate WSRP services without any programming effort. The only effort required is the actual deployment of remote portlets in the local portal server (Hepper & Hesmer, 2003). We are not taking into account the effort needed for the "implementation," that is, the design of the portal page which is needed in any case.

The WSRP specification (WSRP specification version 1, 2003) is the effort of the working group at OASIS (http://www.oasis-open.org/committees/wsrp). It aims to provide a set of options for aggregating user-facing Web services (remote portlets) from multiple remote Web services within one portal application. WSRP standard has been conceived for implementing simple services. The developer of the portlet provides the markup fragments to display Web service data. The current version allows for more complex services that require consumer registration, support complex user interaction, and operate on a transient and persistent state maintained by the service provider. Before looking at the functionality of WSRP, note that what WSRP refers to as a portlet is the combination of a portlet implementation and any configuration data that supports the implementation.

WSRP AND WSRP RELATED STANDARDS

WSRP defines the notion of valid fragments of markup based on the existing markup languages such as HTML, (X)HTML, VoiceXML, cHTML, and so forth. (Figure 1). For markup languages that support CSS (Cascading Style Sheet) style definitions, WSRP also defines a set of standard CSS class names to allow portlets to generate markup using styles that are provided by WSRP compliant portals such that the markup assumes the look and feel of the consuming portal.

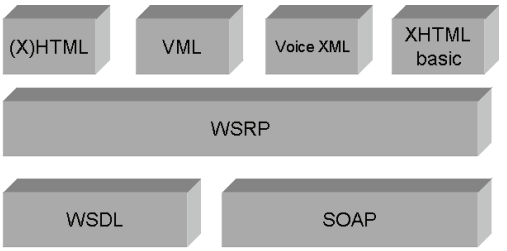
WSRP is fully integrated with the context of the Web services standards stack. It uses WSDL additional elements to formally describe the WSRP service interfaces and requires that at least SOAP binding be available for invocations of WSRP services. WSRP also defines the roles of Web service *producers* and *consumers*. Both *producers* and *consumers* use a standard protocol to provide and consume Web services for user facing portlets. The WSRP specification requires that every *producer* implement two required interfaces, and allows optional implementation of two others:

1. **Service Description Interface (Required):** This interface allows a WSRP *producer* to advertise services and its capabilities to consumers. A WSRP *consumer* can use this interface to query a *producer* to discover what user-facing services the *producer* offers.
2. **Markup Interface (Required):** This interface allows a *consumer* to interact with a remotely running portlet supplied by the *producer*.
3. **Registration Interface (Optional):** This interface serves as a mechanism for opening a dialogue between the *producer* and *consumer* so that they can exchange information about each others' technical capabilities.
4. **Portlet Management Interface (Optional):** This interface gives the *consumer* control over the life cycle methods of the remote portlet.

URL Generation Concept

To support user interaction, all the URLs embedded in the markup fragment returned by the remote *producer* service

Figure 1. Related standard



must point back to the *consumer* application. Therefore, the *consumer* needs to send a URL template as part of the invocation of the getMarkup() method. For example, the consumer may send the URL template with two variables: navigationState and sessionID:

```
http://neptune.monash.edu.au/myApp?ns={navigationStat
e}&si={sessionID}
```

The *producer* responsibility is to generate a markup fragment in which all the interaction URLs must point back to the *consumer*. The *producer* generates a link pointing to the URL replacing the template variables navigationState and sessionID with concrete values:

```
http://neptune.monash.edu.au/myApp?ns=page2&si=4A
HH55A
```

Alternatively, the predetermined pattern allows the *producer* to create URLs that are compliant with this pattern. The *consumer* then parses the markup and rewrites variable parts of URL to point back to the application.

ROLE OF PRODUCERS AND CONSUMERS

WSRP is a protocol in which the interaction always occurs between two Web applications or Web services. The *consumer* application acts as a client to another application called *producer*. The *producer* provides end-user-facing (also called presentation services) Web services in the form of remote portlets. These remote portlets are aggregated into the *consumer's* portal page in the same way as local portlets.

Let's start with comparing WSRP with a Web services application. The Web-based application *consumer* uses HTTP, SOAP, and browsers to interact with remote servers hosting Web services. In response, they receive Web service raw **data** needed to create the markup (typically HTML or HTML form). The input data are posted by submitting the form via a browser.

HTTP protocol is also utilized with WSRP. *Consumers* can be seen as intermediaries that communicate with the WSRP *producers*. *Consumers* gather and aggregate the **markup** delivered by local as well as remote portlets created by the *producers* into a portal page. This portal page is then delivered over SOAP and HTTP to the client machine (PC or a workstation). The *consumer* is responsible for most of the interactions with the remote systems, ensuring user privacy and meeting the security concerns with regard to the processing information flow.

In the sense of additional capabilities, today's *consumers* of WSRP are more sophisticated than simple Web service clients:

1. *Consumer* aggregates multiple interface components (local and remote portlets) into a single page. In addition, features like personalization, customization, and security are also available for remote portlets.
2. The aggregation into a single page is not straightforward because it involves applying *consumer*-specific page layouts, style, and skins to meet the end-user requirements. Therefore, the *consumer* must have knowledge of *presenting* related features in remote portlets to apply customization and rendering.
3. The *consumer* can aggregate content produced by portlets running on remote machines that use different programming environments, like J2EE and .NET.
4. *Consumers* are able to deal with remotely managed sessions and persistent states of WSRP Web services.

The *producer* is responsible for publishing the *service and portlet capabilities descriptions* in some directory, for example, UDDI. It allows the *consumer* to find the service and integrate it into portal. The purpose of the portlet capabilities description is to inform the *consumer* about the features each portlet offers. *Producer's* major responsibilities are listed below:

1. *Producers* are capable of hosting portlets (they can be thought of as portlet containers). Portlets generate markup and process interactions with that markup.
2. *Producers* render markup fragments, which contain Web service data.
3. *Producers* process user interaction requests.
4. *Producers* provide interfaces for self description and portlet management.

The *consumer* can optionally *register* with the *producer*. The *producer* is responsible for specifying whether the registration is required. Typical registration contains two types of data: *capabilities* (for example, window states and modes the *producer's* remote portlets support), and *registration properties* (required data prescribed in the service description). Upon successful registration, the *consumer* receives a