WSRP Producer Architecture

Table of contents

Producer and consumer are provided with a very modular architecture enabling an easy exchange of module implementations. All modules excel by interfaces based on the WSRP object model hiding the runtime environment's (portal's) object model and thus gaining independence of changes in the environment's implementation or design.

WSRP Producer Architecture

1. WSRPEngine

The WSRPEngine is the WSRP implementation endpoint. This class must be deployed in the app server (Tomcat) as a web service. The WSRPEngine implements the WSRP protocol specific ports (=interfaces) and does the corresponding protocol handling. There are four WSRP ports:

Markup
deals with Portlet invocation and Session handling
PortletManagement
covers lifecycle and properties of portlets
Registration
enables a consumer to register at the producer
ServiceDescription
enables a consumer to discover the services that a producer provides

Note:
To be able to reuse Portal functionality regarding session and request handling when invoking a portlet the WSRPEngine must be able to access the HttpServletRequest.

2. ConsumerRegistry

This component manages and provides access to the registered Consumers.

3. HandleGenerator

This component is responsible for generating IDs / handles required for the WSRP protocol handling.

4. WSRP Object Model

The WSRP object model is being generated from the WSRP specification's WSDL types.

5. Provider

The Provider is the access point for the WSRPEngine to the Provider components which hide the provider (portal) implementation's components required to handle and invoke portlets. All Subcomponents wrap corresponding provider components and map the WSRP object model to the provider object model.

6. Portlet Invoker

The PortletInvoker wraps the Provider's invocation mechanisms and provides the Provider with the required environment.

7. Description Handler

The DescriptionHandler manages and provides the description of the provider regarding configuration properties like registration or session handling policy, etc. It moreover provides the descriptions of the provided portlets depending on the registration.

8. PortletPool

The PortletPool manages the portlet instances and is responsible for the portlets' lifecycle management (clone, destroy).

9. Portlet State Manager

The PortletStateMananger enables a Producer to access a portlet's state as a blob that than can be delegate to the consumer to be stored on consumer side.

10. Session Handler

No additional session handling implementation for the Pluto provider is required as the session handling concept is completely HTTP (cookie) based and relies on consumer' cookie handling.

11. URL Composer

The URLComposer must be used for WSRP triggered portlet invocation to create WSRP URLs instead of the portal's URL handling implementation. Therefore the URLComposer is being used by WSRP's version of the DynamicInformationProvider which is being used by the Portlet API implementation to generate portlet URLs. There are two ways how URLs can be composed in a WSRP environment:

1. Via templates that represent URLs that are valid on Consumer side and contain

placeholders for all portlet specific URL components.
2. Via URL rewriting. In that case WSRP specific URLs are being composed that will have to be rewritten by the Consumer.