DocV1.wsrp

WSRP service

The Web Service for Remote Portals (WSRP) is a specification defined by the OASIS group. It allows to query portlets hosted on a remote portlet container. Due to the use of web service (WSDL/SOAP), the deployed portlets and the client portal could be implemented in different language. Indeed, it is possible to use a .Net portal to query portlets hosted on a J2EE portlet container.

The WSRP service is composed of two parts :

- The WSRP consumer : the web service client
- The WSRP producer : the web service server

To generate the Java files associated with the WSDL one we used the wsdl4j package.

- WSRP producer

The producer API is not a Java one but a Web Services one. The WSRP protocol lets you define 4 interfaces to be exposed as Web Services Port. Two of this interfaces are mandatory while two are optional. We have decided, that our implementation will have to support all of them.

It is intended that the implementation of this service will leverage the portlet container one, and will therefore allow JSR 168 portlets to be reached by WSRP consumers.

It is also intended that any implementation of that service will implement the entire stack of optimization defined in the WSRP specifications.

Note that the WSRP protocol is like the JSR 168 one, it defines a 2 phases process :

Protocol	access Business logic	Render Markup
WSRP	performBlockingInteraction()	getMarkup()
Portlet API	processAction()	render()

You can refer to the implementation documentation for that service to find more

information about what tool were used. You will just find here a short presentation of the 4 WSRP Ports.

- Service Description Operations : This interface allows the consumer to get all the necessary information about a producer. Of course this contains the portlet lists and all their meta-data.

```
ServiceDescription = getServiceDescription(RegistrationContext,
desiredLocales )
```

- Registration Operations : some producer, and our implementation is one of those, need a registration from Consumers. Here are the three methods defined by the interface :

```
RegistrationContext = register(RegistrationData);RegistrationState =
modifyRegistration(RegistrationContext, RegistrationData);ReturnAny =
deregister(RegistrationContext);
```

- Markup Operations : this is the interface that allows to access both the portlet business logic and their generated markup. There are four methods in that interface :

```
MarkupResponse = getMarkup(RegistrationContext, PortletContext,
RuntimeContext, UserContext, MarkupParams);BlockingInteractionResponse =
performBlockingInteraction(RegistrationContext, PortletContext,
RuntimeContext, UserContext, MarkupParams, InteractionParams);ReturnAny
= initCookie(RegistrationContext);ReturnAny = releaseSessions
(RegistrationContext, sessionIDs);
```

- Portlet Management Operations : this port introduces some advanced features of the WSRP protocol like cloning a portlet. From the JSR 168 point of view it simply means to associate a new set of preferences to that portlet but with the value of the cloned portlet. Here are the methods

```
PortletDescriptionResponse = getPortletDescription(RegistrationContext,
PortletContext, UserContext, desiredLocales );PortletContext =
clonePortlet(RegistrationContext, PortletContext,
UserContext);DestroyPortletsResponse = destroyPortlets
(RegistrationContext, portletHandles);PortletContext =
setPortletProperties (RegistrationContext, PortletContext, UserContext,
PropertyList);PropertyList = getPortletProperties (RegistrationContext,
PortletContext, UserContext, names);PortletProperties DescriptionRespons
e = getPortletPropertyDescription(RegistrationContext, PortletContext,
UserContext, desiredLocales );
```

Of course the direct access to those interfaces is hidden to you by the eXo portal

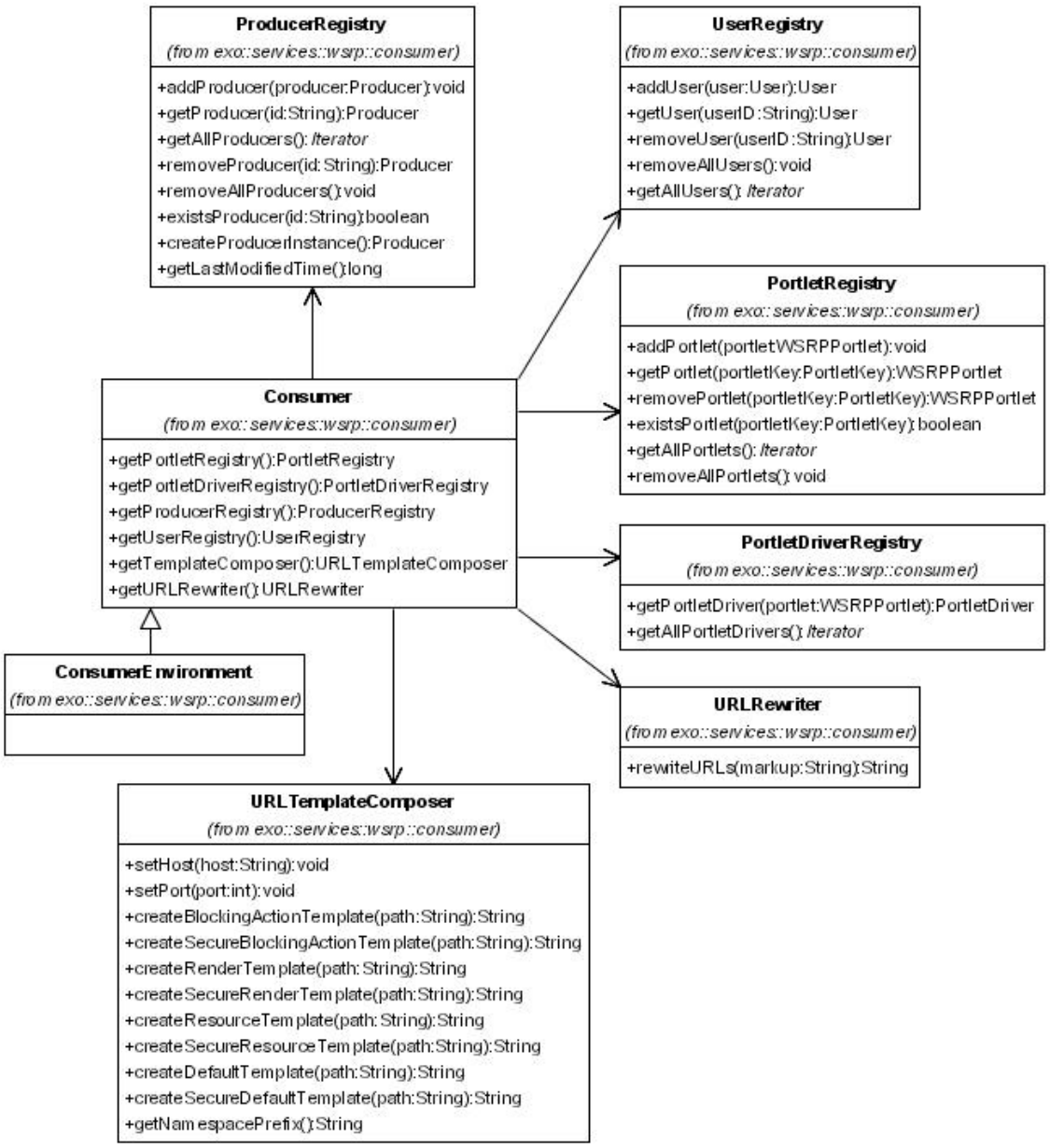
and the eXo platform has implemented a WSRP Consumer portlet which is a JSR 168 portlet that acts as a proxy to any remote WSRP portlet. For more information on that portlet, which uses the Consumer service, just have a look in the portlets related documentation.

- WSRP consumer

The consumer service manages the set of available producers and the information about them. It also abstract the real access to the WSRP layer. The java API used by the eXo platform consumer was originally taken from the WSRP4J project. Some improvements to that API have been made. Note that the implementation has been completely re-writen and once again use of IoC has been introduced.

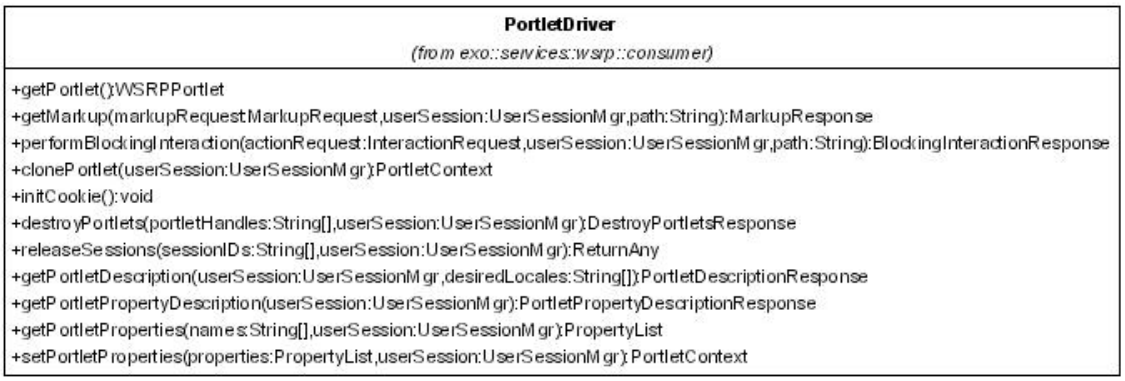
This service is used by our Consumer JSR 168 portlet.

The main entry point is the ConsumerEnvironment class. As usual it can be injected in any component or looked up with the service locator.



This set of interfaces is just a convenient way to sort all the different producers the consumer interacts with. That is why the ConumerEnvironment class is mainly an access to several repositories (ProducerRegistry, UserRegistry, PortletDriverRegistry...).

Once the consumer client (think of our JSR 168 portlet) has looked up the correct PortletDriver object using the several registries, it can interact with the remote WSRP portlet.



Most of the WSRP ports are accessed using the PortletDriver. Only the ProducerRegistry references (via the Producer object) the getServiceDescription() port.

DocV1.wsrp (en)
CrÃ©ateur: XWiki.exo Creation Date: 2005/02/09 22:09
Dernier Auteur: XWiki.exo Last Modification Date: 2005/02/15 23:10