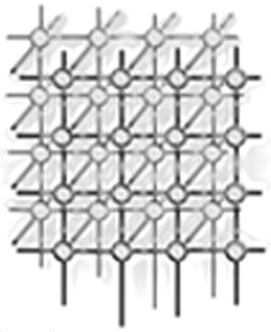
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2005; **00**:1–10 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Investigation of WSRP support in selected open-source portal frameworks



X. Yang*,†, X. D. Wang and R. Allan

CCLRC e-Science Centre, CCLRC Daresbury Laboratory, Warrington, WA4 4AD, U.K.

SUMMARY

WSRP is a promising Web Services specification used to standardise presentation-based access to remote web portal content. In combination with the JSR 168 standard, portlet developers are freed of doing the hard work in publishing their portlets to remote portal – ideally it is the duty of portal framework to publish the portlets it manages. Code-reusability can now be achieved with JSR 168 while "deploy once" is achieved through WSRP. In this paper the WSRP support provided by selected open-source portal frameworks including eXo platform, Liferay, StringBeans and uPortal are reported. WSRP4J, which is not a portal framework but a reference implementation of the WSRP 1.0 specification and the basis of WSRP support in many portal frameworks, is also discussed.

KEY WORDS: WSRP; portal; portlet; web services

1. INTRODUCTION

WSRP [1] and JSR 168 [2] have now become prominent [3, 4, 5] following their emergence in late 2003. The greatest benefits of both standards is reusability of code. This is done firstly by solving interoperability issues between portlets and portlet containers or portal frameworks using JSR 168 and between portal frameworks using the WSRP 1.0 specification. Today more and more open-source portal frameworks are emerging which give portal developers many choices. The focus of these portal frameworks is quite diverse, their markets ranging from education (uPortal [6] for example) to enterprise (such as eXo platform [7] and Liferay [8]). All these portal frameworks provide a JSR 168 implementation together with additional

*Correspondence to: CCLRC e-Science Centre, CCLRC Daresbury Laboratory, Warrington, WA4 4AD, U.K.
†E-mail: x.yang@dl.ac.uk

Received 22 December 2005
Revised ?? ?? 2006

functionalities such as JAAS, JSF, and user friendly administration support. According to our earlier evaluation of some selected open-source portal frameworks [9], the JSR 168 support is quite good for all portal frameworks tested. Migration of JSR 168 portlets between different portlet containers normally involves just some modifications of the configuration files without the source code been touched. The real difference between these portal frameworks lies in the complexity of portlet deployment and management plus additional functions they provide. For example, portlet deployment on eXo platform is quite simple since the container can look for all available portlets automatically while on uPortal a channel needs to be published before it can be subscribed to.

Besides the JSR 168 standard, many portal frameworks nowadays also claim to support WSRP. This could be confusing because of the nature of the WSRP 1.0 specification. Two of the actors defined by the standard are *Producer* and *Consumer*. Producers are presentation-oriented Web services that are modelled as containers of portlets. They should be able to render markup fragments and process user interaction requests. On the other hand, consumers are web service clients that make use of producers to present the generated markup to end users and manage the users' interactions with the markup. When a portal framework claims the WSRP support, it could mean any of the following options:

(i) *Consumer*-only support - Users should be able to consume remote portlets using tools provided by this type of portal frameworks. For instance, uPortal 2.x has WSRP consumer support without a producer;
(ii) *Producer*-only support - WSRP4J [10] producer plus Pluto [11] can be treated as such a portal framework which has the ability to publish its portlets as remote portlets if the consumer is not deployed;
(iii) *Consumer*-and-*Producer* support - Such a portal framework can publish portlets deployed within it as remote portlets and also consume remote portlets using its WSRP consumer. For instance, both eXo platform and Liferay claim this type of WSRP support.

Because of the fuzzy WSRP support claims we decided to carry out an evaluation of the actual support offered by the more popular open-source portal frameworks. This included eXo platform, Liferay, StringBeans [12], uPortal and WSRP4J. WSRP4J is in fact an implementation of the WSRP 1.0 specification with both producer and consumer included. It is not a portal framework but makes use of Pluto to host portlets (a JSR 168 reference implementation by Apache). Results of the investigation are reported in this paper.

2. SELECTED PORTAL FRAMEWORKS AND TEST PORTLETS

2.1. WSRP4J

In WSRP4J, both producer and consumer are provided with a modular architecture which enables an easy exchange of module implementations. The producer comes with a full implementation of four WSRP protocol specific interfaces: ServiceDescription, Markup, Registration (optional) and PortletManagement (optional).

As stated in the WSRP4J documentation, a consumer is used to aggregate the integrated WSRP portlets and forward all invocations together with relevant context and request information to the remote WSRP services. WSRP4J comes with two consumer examples, SwingConsumer and ProxyPortlet. SwingConsumer is a standalone application while ProxyPortlet is a standard JSR 168 portlet. The latter is widely adapted in different portal frameworks to provide WSRP consumer support. For instance, uPortal extends the ProxyPortlet with some minor modifications.

2.2. Selected open-source portal frameworks

Based on our early investigation focusing on JSR 168 support of some open-source portal frameworks [9], eXo platform, Liferay, StringBeans and uPortal are tested for WSRP support. These portlet containers plus GridSphere [13] cover the major open-source portlet containers currently available. GridSphere is not included in this study because there is no WSRP support and the project is now under maintenance stage without further development of WSRP planned in the near future.

eXo platform is a JSR 168 compliant enterprise portal based on JSF, Pico Container, JBossMX and AspectJ. It is defined as a portal and content management system (CMS). Version 1.0 has been released for a while and WSRP support of both producer and consumer are declared in the online Wiki maintained by eXo.

Liferay, another enterprise portal, supports both JSR 168 and WSRP specifications. It comes with a lot of useful features like CMS, single sign-on (SSO) and Aspect-Oriented Programming (AOP). The current version of Liferay is 3.6.1. Both WSRP producer and consumer are included.

StringBeans portal is a standard JSR 168 compliant portlet container but comes with very effective administration portlets for managing the portal Web site. For this reason it was selected to host the UK NGS (National Grid Service) Portal [14]. The latest release of StringBeans v3.0 provides full WSRP support.

uPortal is widely adopted for academic institutions and was designed for educational purposes. It was born even before the WSRP and JSR 168 specifications and had its own definition of "channels" which contributed to the standards. Therefore it is not surprising that uPortal 2.x still has its own mechanism for supporting JSR 168. In fact, the JSR 168 support of uPortal is through channel adaptors to talk to Pluto rather than native portlet support. Version 3, the next release, is supposed to have a new architecture for native portlet support. The current stable release of uPortal is 2.5.1 which has WSRP consumer support. Although not officially announced, uPortal 3.x is expected to have both WSRP producer and consumer. uPortal 2.4.2 has been used throughout our tests because its stability.

2.3. Test portlets

Two standard JSR 168 compliant portlets were selected for our WSRP support test: *HelloWorld* and *LdapBrowser*. As shown in Fig. 1 a), the HelloWorld portlet has a simple user interface with a text input box and a button for submission of the input text. When the button is clicked, some text messages will be displayed with a JavaScript URL link (see Fig. 1 d)). Fig. 1

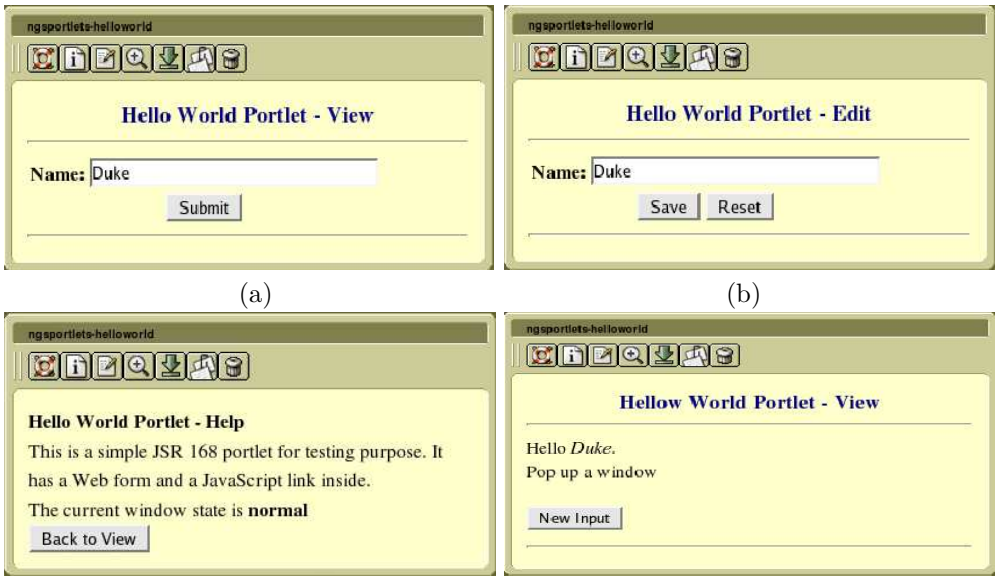


Figure 1. HelloWorld portlet: (a) default *view* mode, (b) *edit* mode, (c) *help* mode, (d) *view* mode, button "Submit" in (a) clicked.

b) and c) are *edit* and *help* modes of the HelloWorld portlet accordingly. The LdapBrowser portlet is not a trial one but deployed in the UK NGS Portal for production usage which is utilised for executing LDAP/MDS query from the NGS MDS server. It provides information such as CPU type and memory amount of the NGS nodes. Similar to the HelloWorld portlet, a web form is used to set the query parameters. Once it is done, the query result will be returned with images and links dynamically set. Further LDAP query can be executed by simply clicking these portlet URL links.

3. WSRP SUPPORT TEST

3.1. Test of HelloWorld portlet

3.1.1. WSRP4J acting as producer

The HelloWorld and LdapBrowser portlets were tested under eXo platform and uPortal locally prior to the WSRP test. Both portlets were then published in WSRP4J which had been downloaded from CVS, built and deployed into Tomcat. The JavaScript link which simply

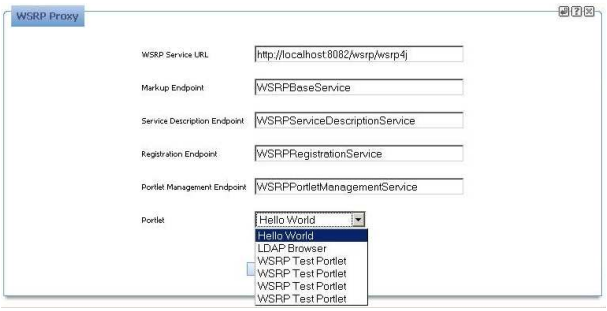


Figure 2. WSRP Proxy portlet in Liferay list remote portlets from a WSRP4J producer.

pops up a window was observed to work but in eXo platform, the portal failed to go back to the original page after the "OK" button was clicked in the popup window. It was first thought to be a problem related to the browser but both Internet Explorer and Firefox had the same issue. It was then observed that uPortal could successfully handle the JavaScript link. Other issues are noted below.

eXo platform 1.0 was observed to consume the HelloWorld portlet without any problem except that there is no *edit* mode support in eXo. With the WSRP4J producer defined in eXo, the WSRP consumer can list available remote portlets. The lack of *edit* mode support however means that users can not set their preferences as they do with local portlets.

Liferay professional 3.6.1 was able to consume this portlet successfully. In Liferay, preference settings are divided into local and remote settings. The local preferences are used to define the four producer interfaces while the remote preferences are for portlet settings which are what users normally do in the *edit* mode. Liferay was also found to be able to retrieve the available remote portlet list automatically from the WSRP4J producer (see Fig. 2). The *edit* support is fully functional in Liferay, but it failed to open the remote help page, displaying an empty page instead suggesting the *help* mode was not fully supported.

StringBeans 3.0 loaded the default page successfully. There were no problem to switch between different modes – *view*, *edit* and *help*. However it was found that no action could be taken within the remote portlet. Each time a button was clicked, StringBeans failed to contact the producer (no SOAP message observed in TcpMonitor) and simply displays an error message. Unlike eXo platform and Liferay, each time a new remote producer is defined the server must be restarted so that StringBeans can load the definition file. But for each remote portlet, StringBeans automatically creates an instance of WSRP proxy portlet, which is then added to the list of available portlets to be added to a page.

uPortal 2.4.2 was able to consume the remote portlet after publishing a WSRP channel. But after either *edit* or *help* mode was entered, it was not possible to go back to *view* mode although in *edit* mode change of preferences were saved correctly when the portlet was accessed the second time. In uPortal, a portlet handle must be provided to publish the WSRP portlet

channel. This is not ideal since the consumer should retrieve such information from the producer rather then it defined by the portal administrator.

3.1.2. eXo platform acting as producer

It was found that another **eXo platform** was able to consume portlets published by an eXo platform. Although with the registration interface implemented, when another eXo acts as a consumer, this interface has to be defined using the portlet management interface.

Liferay failed to talk to the eXo producer. Observing the SOAP message in TcpMonitor shows that Liferay failed to execute the getServiceDescription() operation.

With markupURL and serviceDescriptonURL defined and a registration handle provided (which was extracted from the eXo to eXo SOAP message), **StringBeans** can list remote portlets provided by eXo in its available portlet list. While some portlets such as the HelloWorld portlet can be displyed, others could not be displayed at all. Again no action can be taken within a remote portlet as observed before.

As mentioned above, a portlet handle is needed for **uPortal** to publish a WSRP portlet channel. Several tests have been tried to use the portlet handle extracted from the eXo to eXo SOAP message, but none of them worked so far.

3.1.3. Liferay acting as producer

After some searching on the internet it was found that Liferay has a WSRP producer integrated, which nevertheless seems far from mature.

Another **Liferay** acting as consumer can load the HelloWorld portlet correctly but the button within the page could not be correctly handled and produces an error message indicating *"an unexpected system error occurred"*. Unlike connecting to the WSRP4J producer, the remote portlet list was not displayed correctly. In order to look for the HelloWorld portlet, source code of the page had to be checked. This can be easily solved by setting up the portlet title correctly.

Using the portlet handle hidden in the portlet list (Liferay to Liferay), **uPortal** could load the default view page as well but no action could be performed.

eXo platform failed to retrieve the remote portlet list although a producer was created.

StringBeans had no problem to list the available remote portlets, but as observed no action could be taken, the default page was always displayed.

3.1.4. StringBeans acting as producer

eXo platform was able to load the HelloWorld portlet. Different modes can be switched but no action was completed correctly.

In **Liferay**, the portlet was loaded successfully, but each time when an action was taken, the same page was displayed again. The same thing happened while remote preferences were being set up – the default edit page was loaded again without accepting new values.

Another **StringBeans** was observed to consume the HelloWorld portlet successfully. The only issue was found that sometimes buttons were not correctly handled. For example, in *help*

mode, clicking the button *"Back to View"* did not redirect to *view* mode which worked fine as a local portlet.

Again it was found not possible to define a portlet handle in **uPortal**.

3.2. Test of LdapBrowser portlet

As noted in Section 3.1.3, none of the portal framework consumers could consume the HelloWorld portlet produced by Liferay, so it is omitted as a producer in this section.

3.2.1. WSRP4J acting as producer

Besides the general results found in the HelloWorld portlet test, more results were found from our test with the LdapBrowser portlet from WSRP4J.

eXo platform was found to function properly executing the first LDAP query by clicking the "Query" button. A set of portlet URL links were set in the result page for further query. However such a further query always resulted in error.

StringBeans had the same issue as in the HelloWorld portlet test. The default page was loaded but no action could be taken.

Liferay and **uPortal** were able to consume the portlet but similar to eXo none of them displayed the HTML image link correctly. It was found that this is due to the relative links being used in the markup.

If two remote portlets are put in the same page, in our case, HelloWorld and LdapBrowser, sometimes uPortal could not handle them correctly but displayed the same portlet twice.

In order to fix the HTML image link issue, absolute URLs including the host name and host port of the producer computer must be prefixed. This can be done either on the producer or the consumer side. In our test, we tried it on the consumer side in the uPortal ProxyPortlet. The test was successful (see Fig. 3) but more issues must be considered for it to be a robust solution. If not rewritten by the producer, it should notify the consumer to rewrite the URLs which are defined as *resource* URLs in the WSRP 1.0 specification.

3.2.2. eXo platform acting as producer

Besides the same HTML image link issue, **eXo** failed to handle the portlet URL. That is, a second LDAP query failed when the portlet URL was clicked as the link pointed to the portlet itself.

The same results were observed with **Liferay**, **StringBeans** and **uPortal** as in the HelloWorld portlet test.

3.2.3. StringBeans acting as producer

Another **StringBeans** could consume the portlet. Similar to eXo to eXo, the first query was handled correctly but no further query worked using the portlet URL. Also it was observed that no preferences could be saved.

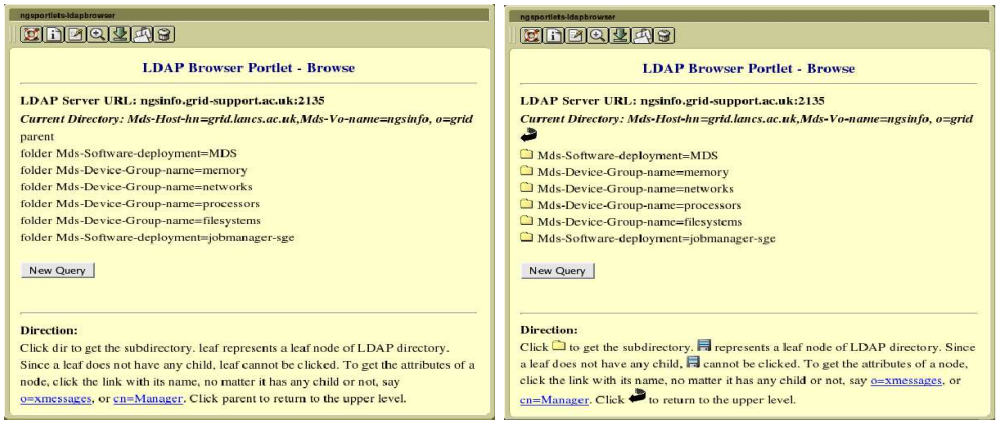


Figure 3. HTML image link - relative links replaced by absolute links.

The same results were observed with **eXo**, **Liferay** and **uPortal** as in the HelloWorld portlet test.

4. CONCLUSIONS AND FUTURE WORK

Traditional web services are data-oriented which means there is no presentation layer for output to be rendered, for example in aggregating applications. As stated in the WSRP 1.0 specification, *"This approach is not well suited to dynamic integration of business applications and content as a plug-and-play solution"*. Therefore the WSRP specification *"defines a web service interface for accessing and interacting with interactive presentation-oriented web services"*. Because of the fuzzy WSRP support claim and lack of existing tests of WSRP in portal frameworkswe undertook this first investigation. The WSRP support is of selected open-source portal frameworks including eXo platform, Liferay, StringBeans, uPortal and WSRP4J has been compared and reported in this paper. We conclude that at present the support provided is not sufficient to make WSRP usable. The reasons are summarised below.

4.1. WSRP producer comparison

(1) A WSRP4J producer is recognised by all of the four portal frameworks selected in this test. Besides the default remote portlet page which loaded successfully, Liferay and uPortal are fully functioned (especially Liferay) and remote preferences can be set correctly. Switching between *edit/help* mode and *view* mode is not possible in uPortal while there is problem with *help* mode support in Liferay. Although the HelloWorld portlet was found to work fine in

eXo platform, eXo failed to handle portlet URLs correctly inside the LdapBrowser portlet. StringBeans could not handle any action in remote portlets.

(2) A producer in eXo platform is only accepted by another eXo although with some careful settings StringBeans has successfully loaded default pages of remote portlets published by eXo. Neither Liferay nor uPortal is able to consume remote portlets in eXo. For uPortal, this is due to the lack of portlet handle which is required while a WSRP channel is published.

(3) Liferay's producer is far from mature although another Liferay, StringBeans and uPortal can connect to it, but none of these created a functional portlet. eXo platform can not set up a connection with the producer at all.

(4) A StringBeans producer is recognised by all portal frameworks selected except uPortal, but none of them created a fully-functional portlet. Another StringBeans can consume the HelloWorld portlet successfully but failed to consume the LdapBrowser portlet since there is a problem with portlet URLs.

(5) Although in the WSRP 1.0 specification only service description and markup interfaces are required, all producers tested have another two optional interfaces, registration and portlet management, implemented.

(6) The HTML image link (*resource* URL) is not correctly handled in any of the portal frameworks. This can be fixed on the consumer side by replacing the relative URLs with absolute URLs by prefixing the host name and port number of the producer machine. This has been successfully tested in uPortal. In practice, the producer should notify the consumer to rewrite such URLs if not done on its own side.

4.2. WSRP consumer comparison

(1) uPortal needs a portlet handle to register a remote portlet which limits its ability to access WSRP producers. When two remote portlets are put in the same page, sometimes uPortal can not handle them correctly and displays the same portlet twice.

(2) eXo platform, Liferay and StringBeans have the ability to retrieve available remote portlets.

(3) eXo platform and StringBeans have a problem handling portlet URLs. StringBeans also has a problem handling buttons with producers other than StringBeans since no SOAP message is observed.

(4) When a new producer is defined in the StringBeans consumer the server must be restarted to connect to the producer which is not convenient to use.

Based on the conclusions above, further development on both WSRP producer and consumer before use of WSRP becomes practical. Although not covered in this test, security is also a big issue in any distributed environment. Therefore our future work will include some further investigation on security, e.g. how to protect remote portlets.

ACKNOWLEDGEMENTS

This work has been undertaken at the CCLRC e-Science Centre supported by UK JISC (The Joint Information Systems Committee).

REFERENCES

1. WSRP Specification 1.0 by OASIS. http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf [1 December 2005].
2. JSR 168: Portal specification. http://www.jcp.org/en/jsr/detail?id=168 [1 December 2005].
3. Castle B. Introduction to Web Services for Remote Portlets. http://www-128.ibm.com/developerworks/webservices/library/ws-wsrp/ [1 December 2005].
4. Vickers S. Portal Standards. *Web Services Journal*, January 2005 18-20.
5. Gupta R.K. WSRP: Dynamic and Real-Time Integration. *Web Services Journal*, August 2005 10-19.
6. uPortal. http://www.uportal.org/. [1 December 2005].
7. eXo platform. http://www.exoplatform.com/. [1 December 2005].
8. Liferay. http://www.liferay.com/. [1 December 2005].
9. Akram A., Chohan D., Wang X. D., Yang X., Allan R. A Service Oriented Architecture for Portals Using Portlets. In *UK e-Science AHM 2005*, Nottingham, UK, 2005; available on CDROM.
10. WSRP4J. http://ws.apache.org/wsrp4j/ [1 December 2005].
11. Pluto. http://portals.apache.org/pluto/. [1 December 2005].
12. StringBeans. http://www.nabh.com/projects/sbportal/. [1 December 2005].
13. GridSphere. http://www.gridsphere.org/. [1 December 2005].
14. Yang X., Chohan D., Wang X. D., Allan R. A Web Portal for the National Grid Service. In *UK e-Science AHM 2005,* Nottingham, UK, 2005; available on CDROM.