Web Services for Remote Portlets – Strengths and Weaknesses Encountered within the European Learning Grid Infrastructure Project

Konrad Wulf
HLRS, University of Stuttgart, Germany
wulf@hlrs.de

Stuart Purdie
DCS, University of St. Andrews, U.K.
sdjp@dcs.st-and.ac.uk

Chris Denham
KMI, Open University, U.K.
C.M.Denham@open.ac.uk

Abstract

Web Services for Remote Portlets (WSRP) has been an OASIS specification since 2003 and basically combines the interoperability and distributed nature of XML Web Services with the modular thin client building blocks approach of the Java Portlet API. In the European Learning Grid Infrastructure project, we have chosen this technology for the application layer of its service oriented architecture and have both experienced the usability of this framework as well as its limitations when it comes to employing the Web 2.0 vision for a smoother user experience and interaction between the Portlets. This became most evident when integrating an instant messaging XMPP web client portlet. In this article we also propose how WSRP could be revised and extended to solve the DOM, client-side-scripting-state and sandboxing problems when using AJAX and how its state-handling could benefit from the OASIS Web Service Resource Framework.

1. Introduction

The OASIS Web Services for Remote Portlets Specification[7] is a standardizing effort for presentation oriented Web Services. In the EU research project European Learning Grid Infrastructure (ELeGI: www.elegi.org) we have chosen this technology for the presentation layer of our open service oriented architecture. We have tried to combine it with AJAX client side scripting, a combination of JavaScript, HTML layers and asynchronous background HTTP requests. In this paper we shall summarize the experiences we have gained so far and propose possible solutions for encountered problems.

2. Motivation

The ELeGI project follows the Open Grid Services Architecture (OGSA [3]) vision and a business model where administratively independent parties federate their resources in order to provide an infrastructure for effective human learning. Such a distributed architecture clearly requires a way to unify and consolidate the building blocks of user interfaces into a single user interface, such as a portal. Therefore the WSRP standard seemed to be the right choice.

As we proceeded it became fairly quickly clear, that this technology alone would not deliver enough means to interact richly when trying to provide the best possible user interaction within a thin web client. Especially with the instant messenger portlet that is required to deliver up-to-date information on people's presences and to exchange messages in real-time mode, it became obvious that the WSRP approach alone would be too heavy and involve unacceptable disruptions of the dialog with the user. In fact, in extreme cases, the portal would not have been usable at all, because of the constant refreshing of the portal's presentation layer.

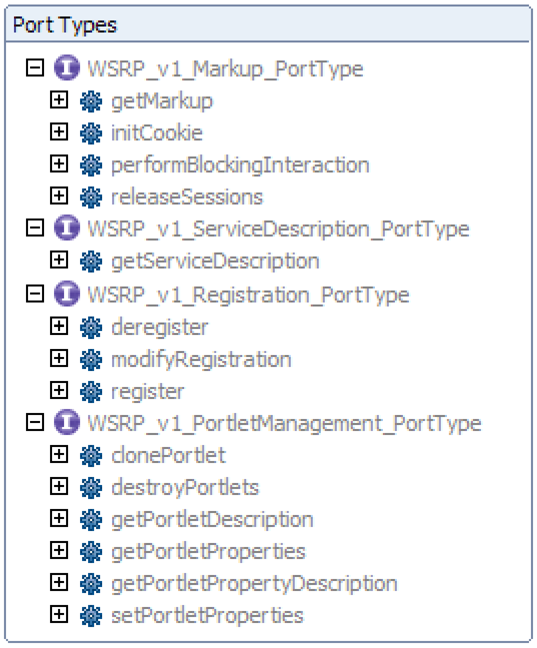
Moreover, it appears to be a ridiculous waste of resources, both with respect to server, client and network load, to refresh a potentially big portal web page whenever a tiny bit of information has been changing. Therefore the ability of client side scripts to make HTTP requests in the background and to precisely only update well-defined parts of the Document Object Model (DOM) tree of an HTML page - as it is the case with AJAX - seemed to be the right supplement to WSRP.

In the following we shall briefly describe both the WSRP standard and the AJAX approach, highlighting some peculiarities, before we then show the problems we have been facing and how we generally propose to solve them.

3. Web Service Remote Portlets highlights

The Web Services for Remote Portlets specification describes 4 port types, two of which are compulsory: the Service Description and the Markup port type.

Optional are the Registration and the Portlet Management port type (cf. Figure 1). Each of them shall be outlined in the following.



Port Types

- WSRP_v1_Markup_PortType
  - getMarkup
  - initCookie
  - performBlockingInteraction
  - releaseSessions
- WSRP_v1_ServiceDescription_PortType
  - getServiceDescription
- WSRP_v1_Registration_PortType
  - deregister
  - modifyRegistration
  - register
- WSRP_v1_PortletManagement_PortType
  - clonePortlet
  - destroyPortlets
  - getPortletDescription
  - getPortletProperties
  - getPortletPropertyDescription
  - setPortletProperties

Figure 1. Port types and methods of the WSRP standard

Service Description: provides the metadata, portlet handles and tells about whether the service requires registration.

Markup: This port actually delivers the HTML mark-up. In order to do this it needs to rewrite URLs and supports the brokering of sessions and cookies. The main method `getMarkup()` requires as input a number of context objects: a `RegistrationContext`, a `PortletContext`, a `RuntimeContext` and a `Usercontext.` In return it sends back a `MarkupContext` and a `SessionContext.`

Registration: Allows WSRP Consumers to register and deregister. If successfully registered, the consumer gets a unique registration handle assigned.

Portlet Management: Provides access to portlet properties for further customization.

4. AJAX highlights

Asynchronous JavaScript And XML (AJAX) is a set of technologies used to provide incremental updates to, and/or return data from a web browser to the server. At the core is the `XMLHTTPRequest` capability, supported by most modern web browsers, that allows a small packet of data to be retrieved by the web browser. This communication occurs in the background, and calls some JavaScript method when data arrives back at the web browser.

By combining this communication channel with modification of the existing HTML web page (via DOM manipulation), a totally dynamic user experience can be created.

5. Main Problems encountered and possible solutions

In the following we shall enumerate major challenges we have faced so far when trying to use WSRP Portlets with AJAX client side scripting. Additionally, some solutions will be proposed.

5.1. Reset of client side state

Writing AJAX code for WSRP requires a certain style of writing. All state changes on the client side should be reflected in cookies. Otherwise the client side state of a portlet gets lost when the portal web page is being rebuild or refreshed. Nevertheless it is worth mentioning that if all Portlets within a given portal would adhere to the AJAX philosophy, whole page rebuilds would not occur and thus this problem would not exist.

5.2. Element naming conflicts in the DOM tree

In order to address a specific HTML element, one can go down the DOM tree or access it by a unique ID that has been declared before. Normally, the uniqueness of each ID can be guaranteed by the page author, as they have total control. With a portal aggregating multiple Portlets, this level of control is not possible, and some solution must be found to ensure that the same ID is not reused.

It is not possible to adopt some convention, such as each portlet author using a domain name they control as a prefix. The major problem with this approach is that it assumes that there will only be one instance of a particular portal on a single page; which is not guaranteed (indeed, for Portlets such as an RSS reader, multiple instances are positively the norm, rather than the exception.)

At first we were concerned about HTML elements and JavaScript variables not being unique within their scope. But after digging deeper into WSRP, we have realized that this has been thought of already:

Within the Java Portlet API, JSR-168, there is a method in the `PortletRequest` class called `getNamespace()`, which gives a guaranteed unique prefix, to be prepended to some portlet specific name. Within the frame of WSRP, this efficiently maps onto the `namespacePrefix` from the `RuntimeContext`. That is, there exists a mechanism for the portal to establish a suitable namespace. Since it is up to each WSRP consumer who has full control of a portal as a whole, this consumer can independently of the producer steer the uniqueness of the prepended name qualifiers.

5.3. Sandboxing problems with XMLHttpRequests

In principle, the same thing for AJAX based requests (`XMLHTTPRequests`) applies as to HTTP cookies, applets and so on: It is considered a high security risk, if clients were to allow a loaded web page to contact any other server than the one where the page originated from (sandboxing, cf. Figure 2). Although there is no explicit standardization of the `XMLHTTPRequest` object in JavaScript, the most commonly used browsers do this sandboxing for normal web pages. This finding has been confirmed by others [5]. Moreover, there is a W3C draft for the `XMLHTTPRequest` object also requiring sandboxing [6].

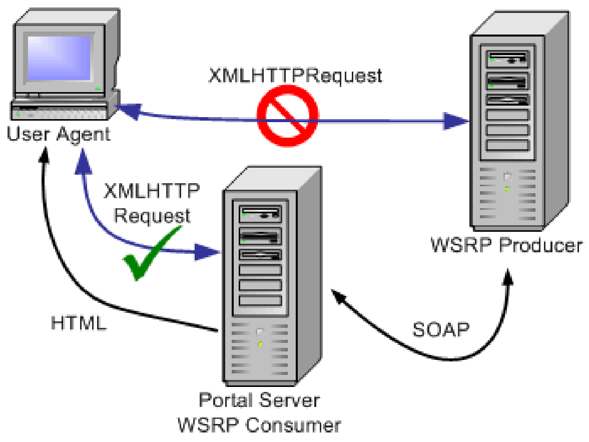


Figure 2. AJAX requests are only allowed to the same server as generated the initial web page

There are two general possibilities to circumvent this behaviour [9]: a. to sign the code with a certificate or b. to proxy the requests. The latter is exactly what has been done for cookies in the WSRP specification, as depicted in Figure 3.

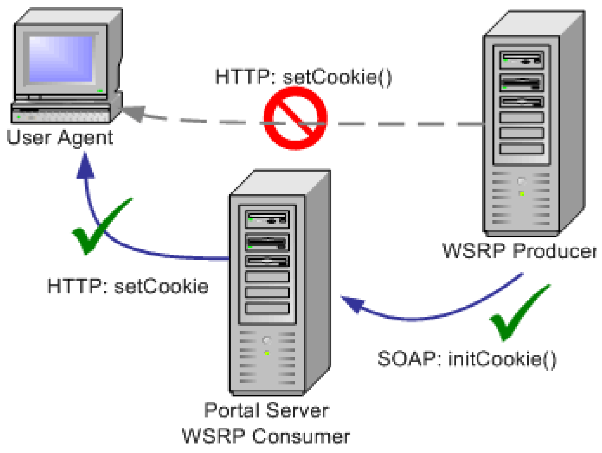


Figure 3. For HTTP Cookies, the WSRP consumer can act as a proxy.

While the same proxy mechanism would be advisable for server functionality that shall be accessed via an `XMLHTTPRequest`, WSRP as it stands fails to provide it. We recommend to add a method `proxyXMLHTTPRequests()` to the Markup port type for that and use the same mechanisms for URL rewriting in the client side code as for regular URLs.

In fact, the other alternative, signing the JavaScript, would not even work in WSRP, because the JavaScript would need to be manipulated by the Consumer for namespace prefixing, after the code has been signed by the Producer. Although not mentioned in the draft standard, most web browsers will allow signed JavaScript code to break out of the sandbox, if the user agrees. The signing prevents injection of hostile code into the JavaScript. However, this signing also prevents any modification, including namespace rewriting. A signing approach would require that the WSRP Consumer signed the code after namespace isolation – requiring support in the Consumer, and the certificates and private keys to be made available to the consumer. That would require a significant degree of trust in the Consumer, beyond what WSRP currently expects and beyond what partnering institutions usually exchange.

For the instant messaging client [8], we have moreover been doing a “trick” to add a notification capability to the IM Portlet. On receiving a HTTP request, the server holds back the response until something has changed on the XMPP server (proxied by the Consumer, naturally). Thus we achieved a kind of “lazy” polling, which is much more efficient than regular polling. This “trick” would be a generic solution for a useful notification capability of Portlets. So if this mechanism would be officially specified in the WSRP spec, a next step towards richer user interaction in a portal could be achieved.

5.4. Single Sign On and WSRP

As soon as a WSRP portal intends to provide personalized portlet content to the end user, the question of single sign on arises. The WSRP specification explicitly states that authentication is out of scope of the specification and that it should be established by other means, such as using WS-Security, SSL or HTTP. This is a problem because the choice of authentication methods is not stated in a WSRP Portlets producer's service description (WSDL). Therefore this needs to be communicated out of band and the inclusion of such an identity-based portlet cannot be done automatically, just by using the information provided in the WSDL.

In ELeGI, we have chosen HTTP authentication, with SSL client authentication as a next step as soon as a public key infrastructure has been established. WS-Security has proven to be a bad choice for interactive use, because of its huge messaging overhead, even when used in conjunction with WS-SecureConversation. Moreover, such message level security instead of transport layer security is really only needed when you have multi-point routed SOAP messages.

WSRP's registration port type should not be confused with a user authentication mechanism: A registration is per WSRP consumer, usually an organization providing a portal to their users, and not per single user.

5.5. WSRP approach to state handling compared to WS-Resource

It is worth noting, that in WSRP each Portlet is not accessed as an extra resource, having a separate URI. Instead, one service usually provides a multitude of Portlets.

WSRP has its own mechanism of handling stateful objects within a web service framework. However, it is not related to the WS-Resource framework [4], which gives a simple, and generic mechanism for doing this. This is not a surprise, as WSRP predates WS-Resource by two years.

There is some impedance mismatch between the two – in WS-Resource, a Resource like a single Portlet can be identified by a single `EndPointReference` – essentially a URL and some arbitrary XML elements packaged together. For WSRP, to address a single Portlet requires the URL, a registration handle, and the name of the desired portlet; all handled separately.

With attempts to use a Portlet to provide some user interface for a WS-Resource (in this case, a Grid service), this mismatch resulted in significant work to handle.

6. Conclusion

In this article we have outlined why it is reasonable to combine WSRP with AJAX, what the challenges are and what should be done to extend WSRP to deliver richer user interaction and single-sign-on in a more automated way. Moreover, it has been shown why it would be beneficial for the WSRP standard to reuse the WS-Resource standard for state-handling.

In summary, we can say that:

- While the maintenance of client-side state is not directly supported by WSRP, a Cookie-oriented programming style will help avoid the destruction of client-side state on rebuilds of the portal web page.

- We have anticipated a problem of namespaces in the DOM tree and JavaScript variables in aggregated web page from various independent sources. This has turned out to be satisfactorily solved within WSRP v1.0 through a consumer controlled prefixing of namespaces.

- The sandboxing of `XMLHTTPRequest` is indeed a problem not satisfactorily solved in WSRP, version 1.0. A sensible way to solve this is to proxy through the Consumer, like it has been done for cookies.

- If, additionally to the proxying of `XMLHTTPRequests`, WSRP formalized the use of the "lazy" polling mechanism described in this article, WSRP would benefit from a generic notification capability in terms of richer user interaction possibilities.

- Which authentication mechanism to be used with the Producer, if any, should also be specified in the Producer's WSDL. But this is a problem of the WSDL specification in the first place.

- WS-Resource is a convincing approach to state-handling. Future versions of WSRP should built on it, even if backwards compatibility with the current version may not be feasible.

7. References

[1] Allamaraju, S. & Brooks, R. (2004) Web Services for Remote Portlets 1.0 Primer [online], Available from: http://www.oasis-open.org/committees/download.php/10539/wsrp-primer-1.0.html (Accessed 26 June 2006)

[2] Castle, B. (2005) Introduction to Web Services for Remote Portlets [online], IBM developerWorks, Available from: http://www-128.ibm.com/developerworks/library/ws-wsrp/ (Accessed 26 June 2006)

[3] Foster, I., Kishimoto, H. & Savva, A. (2005) The Open Grid Services Architecture, Version 1.0 [online], Available from:

http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf (Accessed 26 June 2006)

[4] Graham, S., Karmarkar, A., Mischkinsky, J. et al. (2006) Web Services Resource 1.2 (WS-Resource) [online], OASIS, Available from: http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf (Accessed June 28 2006)

[5] Herrington, J. (2006) Ajax RSS reader - Use Ajax to build an RSS reader [online], Available from: http://www-128.ibm.com/developerworks/xml/library/x-ajaxrss/ (Accessed 26 June 2006)

[6] Kesteren, A (in press) The XMLHttpRequest Object, W3C Working Draft. Available from: http://www.w3.org/TR/XMLHttpRequest/ (Accessed 27 June 2006)

[7] Kropp, A., Leue, C. & Thompson, R. (2003) Web Services for Remote Portlets Specification - Version 1.0 [online], OASIS, Available from: http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf (Accessed 26 June 2006)

[8] Open University (2006). MSG – User Guide. Available from: http://msg.open.ac.uk/userguide/ (Accessed 28 June 2006)

[9] Snell, J. (2005) Call SOAP Web services with Ajax, Part 1: Build the Web services client [online], IBM developerWorks, Available from: http://www-128.ibm.com/developerworks/webservices/library/ws-wsajax/ (Accessed 26 June 2006)