Portlet Feasibility Study

A report prepared for the GroupLog Project funded by JISC under the eTools Programme

Document details

Authors:	Monica Duke (UKOLN) and Elaine Swift (CDNTL)
Date:	29th April 2005
Version:	1.0.
Document Name:	groupLog-portlet-feasibility.pdf
Notes:	Submitted to JISC and published on the Web at http://www.bath.ac.uk/e-learning/grouplog/jisc/

Summary

This is a report on a study about the feasibility of developing the GroupLog application as a WSRP portlet, funded by JISC as part of the eLearning Tools for Learners and Teachers Programme 3/04.

Acknowledgements

UKOLN is funded by the Museums, Libraries & Archives Council (MLA), the Joint Information Systems Committee (JISC) of the Higher Education Funding Councils, as well as by project funding from the JISC and the European Union. UKOLN also receives support from the University of Bath where it is based.

The GroupLog project was lead by the Centre for the Development of New Technologies in Learning (CDNTL), based at the University of Bath

Primary Audience

This study was prepared between September 2004 and March 2005 for the JISC eTools GroupLog project, to investigate the feasibility of offering GroupLog as an interface or organisation of Web Services that could be utilised within a portal. Specifically, the study reports on the feasibility of porting GroupLog over to a portlet model according to WSRP specifications.

The study therefore describes the protocols and systems required to achieve that aim, in technical terms. A description is provided of the development work undertaken to illustrate the issues concerned, with code samples. The report summarises the maturity of the standards and the availability of resources, and indicates the technical effort and skills required and different approaches that could be adopted by GroupLog. Some general conclusions are drawn, so that together with the background knowledge provided, the technical lead and the project manager can make decisions regarding the options available to GroupLog, and the feasibility of those options.

General Audience

We believe that the study will also be of interest more generally to the JISC community:

Decision Makers

Decision makers may find that the technical nature of the text required in some sections of the report are hard to engage with directly. To facilitate the reading of the report, the non-technical audience is advised to consult some other documents first, depending on their prior knowledge in the area of portals. The following documents are suggested:

A literature review that aims to provide readers new to institutional portals with an introduction to the topic and an overview of outputs from a number of institutional portal activities. [1]

The JISC Portals FAQ [2]

An Ariadne article that gently introduces JSR168 and WSRP in the context of CREE, another JISC-funded project. [3]

An introduction to WSRP [4]

An introduction to the Java Portlet Specification (JSR168) [5]

Additionally, a small glossary has been provided in Appendix A to familiarise the reader with the main terms that will be encountered. This particular audience may wish to read selectively, omitting section 6 and parts of section 7.

Technical developers

Technical developers will find a large number of links to resources that provide entry points for those considering undertaking work in the area of portals and WSRP. The analysis summarises the state of the art at the time of writing. The examples show how WSRP development can work in practice.

1 Executive Summary

GroupLog is a project funded by the JISC under the 03/04 call for projects to develop E-Learning Tools for learners and teachers. This is a report on an assessment of the feasibility of developing GroupLog as a WSRP application, carried out by UKOLN in conjunction with CDNTL. Web Services for Remote Portlets (WSRP) is first reviewed as a standard by covering the background information obtained from desk research. The review contains extensive links to portlet-related resources. Development options are then considered. A test platform that was installed to complement the desk-research by providing hands-on experience is then described, together with a sample application illustrating two different ways in which it could be delivered by WSRP. This report concludes that a solution that involves the use of Java-related technology is the most practical option given the current status of the technology. This approach would require a suitable Java platform to be set up and configured. Significant technical knowledge and programming effort would be required. The time and experience required to implement such a solution does not fit within the immediate plans and timing for GroupLog development. The report ends by drawing some general conclusions that can be used to inform any future decisions regarding the delivery of GroupLog using WSRP.

2 Contents

3 Introduction

This report was written as part of the Joint Information Services Committee (JISC) eTools programme[1] and forms part of the GroupLog project. The GroupLog project[2] is currently developing a web-based tool that facilitates collaborative group working within large student cohorts.

JISC, in conjunction with Industry Canada, the Australian Department of Education, Science and Training and the US Advanced Distributed Learning Initiatives, are currently describing common e-learning frameworks which have a coherent, common vision for how future development of e-learning services and applications should progress. The framework draws upon experiences of other organisations such as MIT Open Knowledge Initiative and, as such, proposes that the framework be based around a service-oriented architecture.

In looking to the future for the GroupLog project and how it may be incorporated within a framework one must consider possible ways that such a framework could be implemented and, thus, possible implications for the development of GroupLog.

One possible way that the e-learning framework could be achieved is via portal technologies through the use of WSRP-enabled portlets i.e. portlets that use the Web Services for Remote Portlets (WSRP) standard. A service oriented approach is commonly implemented using SOAP-based Web services. There is thus a natural synergy between using SOAP to deliver the component services that make up an e-learning application and using the Web Services for Remote Portlets (WSRP) standard to embed the user-facing parts of the application into third-party portals.

This report investigates the feasibility of developing GroupLog as a WSRP application by looking into the technology further, highlighting any potential risks, issues and advantages that need to be considered in the general development of WSRP-enabled portlets. It then looks at the prototype development of the Resource Discovery Network's ResourceFinder as a possible model for the development of GroupLog and then finally discusses the feasibility of implementing a portlet version of GroupLog.

4 Background Information: Service Orientated Architectures

4.1 JISC E-Learning Framework

The JISC E-Learning Framework (ELF)[3] describes several layers of components that are considered important in the compilation of a system that supports e-learning. The framework identifies three different layers of functionality; common services, learning domain services and user-agents. (See the ELF web page for a full diagram). Into these three layers are placed different components, or services, that have been identified as required to support the layers above.

[1] The JISC eTools Programme http://www.jisc.ac.uk/elearning_tools_home.html
[2] The GroupLog Project Home Page http://www.bath.ac.uk/e-learning/grouplog
[3] The ELF http://www.elframework.org/

By extracting out components that are common to many e-learning applications it is envisaged that applications utilised within such a framework could employ those common services rather than each application employing its own version of the service thus reducing duplication of effort and increasing the ease with which a large loosely couple learning environment could be implemented. A framework that utilises these concepts is said to be employing a Service-Oriented Architecture (SOA).

As Wilson et al [6] explain in their paper, the framework is merely a description of the possible components within the framework and it neither describes a design nor an implementation of the framework.

In outlining some examples of framework design and possible implementations, Wilson et al demonstrate how such an architecture, for educational systems, can be of benefit. For example, they propose an e-learning framework that shares components with an e-Science framework and suggests that one possible implementation could employ the use of portals to utilise some of the identified web services.

It should be noted, however, that the ELF is a proposal for how future e-learning systems could be developed rather than the present developmental state of e-learning within the UK or abroad. At present, very few of the services identified by the framework exist in a robust form though JISC are presently funding programmes and projects that will develop some of these services. Furthermore, as part of this report will illustrate, some of the technologies, upon which the framework is basing development of these services, are still in their infancy and hence a full implementation of an e-learning system that is based on the JISC e-learning framework is still some time away.

4.2 Web Services

A SOA differs from a standard application architecture in that it relies upon functionality for an application being provided by discrete pieces of software which are not necessarily embedded within the application itself. These pieces of functionality are collectively called services and may reside in another application or be entirely standalone. This concept can be extrapolated further to the extent that the services may be distributed among servers that reside in different organisations and may be accessed via the Internet. Thus a Web Service can be described as a piece of functionality, exposed through a network such as the Internet, which uses an agreed set of standards to exchange data with other Web applications.

The World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS) have been fundamental in refining the agreed standards via which Web Services can communicate with each other and other applications.

Firstly, a Web Service exchanges data with applications using XML (Extensible Markup Language). The XML is structured according to a standard protocol called Simple Object Access Protocol (SOAP). It is this structured XML document that an application using a Web Service can then process and utilise, displaying the data to the user as the internal application logic determines.

To use a Web Service an application has to know what functionality is available and how it may use that service. This information is published using another standard, the Web Service Description Language (WSDL), which describes, again in XML, how to communicate with the desired Web Service.

To enable applications to discover Web Services, their information may be published in a 'service registry' using Universal Description, Discovery, and Integration (UDDI), an XML-based system for registering Web Services, i.e. a yellow pages of Web Services.

The XML data is typically transported between an application and a Web Service via the Internet either using standard protocols (HTTP) or encrypted (SSL). While SOAP is the more versatile protocol to use in consuming Web sSrvices an alternative method of calling a Web Service is through the use of a REST (Representational State Transfer) style URL. This URL has calls to functions available embedded in the URL itself rather than being passed via a SOAP XML file and is limited to using HTTP as a transport mechanism. Amazon.com have exposed some of the functionality of their Web sites as Web Services and applications wishing to use them send requests via a REST-style URL.

A useful example, illustrating the concept of consuming web services, can be found at http://www.allconsuming.net/ where the author utilises the web services provided by Amazon to display information regarding books that have been recommended by authors in their weblogs.

4.3 Portals and Portlets

A portal is a Web-based application that offers a suite of commonly used tools and serves as a single point of access to these tools. It is sometimes, but not always, possible to personalise elements of a portal. Within the e-learning community, an e-learning portal can offer both students and staff access to a variety of required services such courses, collaborative toosl, library facilities or student administration. In essence a portal can be considered a one-stop shop to all the services that a particular community requires. Further background information regarding portals can be found at [2].

A portal can be either built from scratch based around an institution's present information architecture or based upon an existing portal framework either commercial or open source.

There is a variety of developmental work around portals within UK Higher Education institutions. The University of Bristol Information Services web site on portals and portal frameworks presents a table[4] of the variety of portals/portal frameworks that are currently being employed by various institutions. From the table it can be seen that there is a large spectrum of portal technologies being employed from ASP.NET or other Microsoft technologies to IBM Websphere (based upon Java) through to open source solutions such as UPortal or Zope. Some institutions are also using solutions

[4] http://www.bristol.ac.uk/is/projects/portal/portalbytes#list

from commercial Virtual Learning Environment providers such as Blackboard (Blackboard Community Portal System) or WebCT (WebCT Vista).

To gain some level of the use of portals in an international arena, uPortal lists over 70 institutions that are using its portal server 'in production' circumstances[5] with another 60 institutions presently implementing the server.

As previously stated the JISC ELF is a description of the possible services that could be incorporated into an e-learning framework rather than an implementation of a system. Utilising the functionality of a portal framework is one way in which the ELF could be realised. In particular, portals can employ discrete blocks of functionality called portlets.

JISC define a portlet as "*In the context of personalisation and embedding, portals can achieve this through creating distinct building blocks of functionality, e.g. cross-search, alerting, listing, and each one offering a visible component to the user. Each building block is known as a portlet. These can be joined together to create a portal environment, within which various degrees of personalisation can be incorporated, or embedded within a separate environment as required. Portlets feature heavily in many of the current portal building frameworks such as the Apache Jetspeed project, IBM's WebSphere Portal Server and Oracle's Application Server Portal*." [2]

Thus a portal can be envisaged as an amalgamation of various portlets.

A portlet provides fragments of HTML code that wrap around the content that has been requested by the portlet container, usually the portal which is then displayed to the user in a single web page together with the HTML fragments from other portlets.

To ensure that portlets and portals can communicate with each other successfully, a Java specification called JSR 168 has been introduced in recent years. Presently not all portals can understand or communicate using this specification but portal developers are currently implementing or looking to implement this specification with their products.

Hence one can see how portlets could be used to deliver some of the individual components that are outlined within the ELF framework. For example, a portal could utilise portlets that supply authorisation, authentication, group management and chat facilities in conjunction with other services such as activity authoring and management.

However, some portlet-development approaches rely heavily on close integration with the portal that contains them. To achieve a fully service-oriented approach as outlined by the JISC ELF, it should be possible to re-use remote portlets, based on independent external services, within a portal.

The remainder of this report investigates this possibility in further detail using the RDN as a case study and then, drawing upon the findings of that study, looks into the feasibility of utilising GroupLog as a WSRP application.

[5] http://mis105.mis.udel.edu/ja-sig/uportal/

5 WSRP – Web Services for Remote Portlets

Web Services for Remote Portlets (WSRP) is a specification approved by the Organization for the Advancement of Structured Information Standards (OASIS). [7] Version 1.0 of the standard was approved in August 2003 and developed through the joint efforts of two OASIS technical committees. According to the specification, WSRP "defines a web service interface for accessing and interacting with interactive presentation-oriented web services."

Portals and other Web applications render and aggregate information from different sources and provide it in a compact and easily consumable form to an end-user. The WSRP specification provides a common protocol and a set of interfaces for presentation-oriented web services to allow easy aggregation in the form of a plug and play solution. The portal acts as an intermediary between end users and WSRP services and aggregates services from many different content providers. The use of WSRP is intended to remove the requirement for significant custom programming effort when integrating remote applications; the writing of special adapters for applications and content providers to accommodate the variety of interfaces and protocols used by content providers is no longer required.

In other words, a portal brings together different information sources and manages their coherent presentation to an end-user. WSRP belongs at the back-end of the portal, providing a standardised way for the portal to access those content sources that it wishes to present to the user. WSRP offers a web-services conformant interface for interaction with content, producing output that is geared towards presentation in portals, and amenable to content management control that is required within a portal co-ordinating content from different sources. The WSRP protocol governs the interaction between the portlet and the consuming portal, and not the implementation specifics of the portlet, therefore local control within the portlet is preserved.

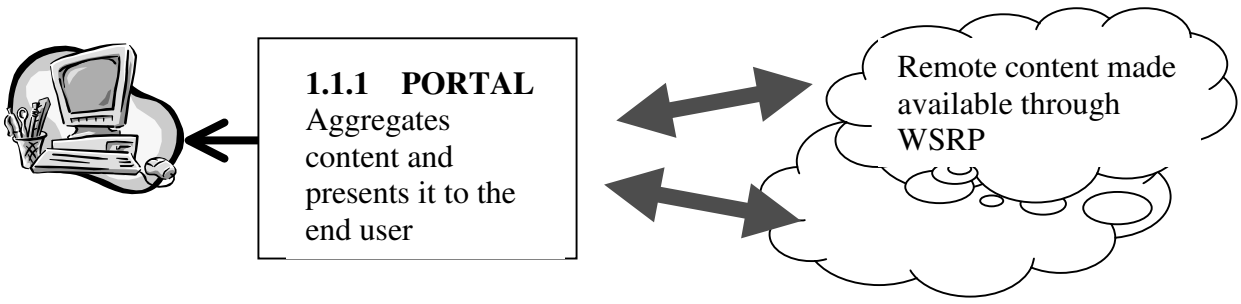

Figure 1: Portals act as user-facing intermediaries that can re-use remote content

For the content-developer, the attraction of WSRP is to "write-once deploy anywhere"; once content is available as a WSRP portlet, it is then deployable in all portals which support the standard. The benefits that are expected to accrue as WSRP products become deployed and WSRP is used to access remote content are [4]:

- Content sources exposed using WSRP will find larger audiences as they are available to add to a larger set of portal pages

- Content sources will be able to better manage the deployment and upgrading of their service by retaining direct control over those processes.

WSRP layers on top of the existing web services stack, including SOAP, XML and URI/URL. The interfaces are defined using Web Services Description Language (WSDL). Future versions are intended to use future applicable Web Service standards such as those that are expected to develop in the areas of Security and Policy.

5.1 Maturity of the WSRP standard

5.1.1 The industry perspective

The Web Services Roadmap website[6], which is part of the CBDI Journal (Insight for Web Services and Software Component Practice), is sponsored by a number of commercial players (e.g. IBM and Microsoft). It provides a useful overview of Web Service related protocols (including the bodies involved in their development and current status): http://roadmap.cbdiforum.com/reports/protocols/summary.php

The site also assesses the status of various Web Service protocols and suggests a timeline for their adoption and relevant roadmap actions. WSRP is classified as an early adoption protocol, defined as "More robust implementations available and protocol well into standards process, encourages production usage by end user organizations". This has been its classification since May 2004. Although WSRP is not as yet considered to be 'mainstream', its use is encouraged.

5.1.2 Implementation availability and support

A measure of WSRP protocol maturity can be made against the quantity of implementations on the market (commercial or otherwise) and the quality of support available to developers wishing to adopt the standard.

5.1.2.1 Implementations

The implementations of toolkits supporting WSRP development are summarised in the next section, and described in more detail in Appendix B. Although there are a number of free and commercial applications, there is a heavy emphasis on Java, and some languages (for example Perl and PHP) appear to be under-represented or not represented at all, particularly when looking for open source and free access to software.

5.1.2.2 Support

Support for WSRP developers revolves mainly around mailing lists, Weblogs, and vendor (or developer) support for a specific product. Weblogs and mailing lists provide the backbone of community support. Product-specific mailing lists although geared to answer product-related queries, also often discuss pros and cons of different approaches and these more general discussions are of wider interest and worth pursuing. Examples of product-specific community support forums include those provided by JBoss[7], LifeRay[8], Exo[9], WebSphere[10] and Oracle[11].

[6] http://roadmap.cbdiforum.com/reports/protocols/

[7] http://www.jboss.com/index.html?module=bb&op=viewforum&f=205

"A Day in the Life of a Software Developer" [8] provides extensive portal and portlet development coverage including polls on the popularity of products (both commercial and open source), news about product development, pointers to articles and resources, as well as opinion. The back-chat (or comments posted in response to the main post) add to the value by balancing the range of views. There is an associated mailing list [9] with over 2000 subscribers (as of February 2005). The same blog is also available at two alternative sites, http://jroller.com/page/portlets (with links to other blogs) and at http://portlets.blogspot.com/. Uncommented bytes [10] contains some portlet-development entries and news amongst other technology-related posts.

A complementary approach to the blogs is the Java.net portlet community site [11], which includes a WIKI with links and specifications, announcements and tips, with a focus on JSR168. It also hosts another portlet blog [12].

A number of vendors (or sometimes independent developers) provide articles introducing the standard, often backed by introductory development on a specific platform. Once again, although focussed on only one product and thus requiring access to that product to follow the examples through to implementation, these articles often address more general points, such as explanations of the flow of processes or guidelines regarding coding style and practice. The latter tends of necessity to be tied to a specific development platform (e.g. Java), and often comes with sample annotated code. Frequently the source code of the example applications is also available for download.

An introductory place to start reading (besides the specifications) is an early article by two of the members of the technical committee that defined WSRP [4] which provides an overview of the main ideas of WSRP. Introducing the Portlet Specification (Parts 1 and 2) [13,14] are JSR168-specific but also discuss alignment with WSRP. An example portlet and further explanations on the relationships between portals, portlet containers and portlets are available in Part 2 of the article[12]. Similar ground is covered in [15] using a different example demonstrated for use with the Liferay product. The code samples (which are in Java/JSP and can be downloaded) are walked through to illustrate the portlet lifecycle and interaction with the portal.

BEAWeblogic's site 'Building Portal Applications' [16] section covers the development of portlets with a focus on using the products' portlet creation wizard, but also contains examples of JSR168 compliant code of configuration files when discussing the production of such code. Sun's "Building JSR 168-Compliant Portlets with Sun Java Studio Enterprise" [17] discusses WSRP and JSR168 and their features. The example (which accesses the Google Web Service) closely follows use of the GUI for portlet creation and deployment, but the code samples are then

---

[8] http://forums.liferay.com/

[9] http://www.exoplatform.com/portal/faces/public/exo/home/community/forum

[10] http://www-106.ibm.com/developerworks/forums/dw_forum.jsp?forum=168&cat=9

[11] http://www.oracle.com/technology/products/ias/portal/discussion_forums.html

[12] Note that an updated version of the code found on the on-line site was distributed on the Pluto mailing list on 13 October 2003 Available at http://nagoya.apache.org/eyebrowse/ReadMsg?listName=pluto-user@portals.apache.org&msgId=1720059

discussed. IBM describes the main issues in converting a portlet from the proprietary IBM Portlet API to the JSR 168 API, highlighting aspects of JSR168 in the process [18]. A sample portlet is available for download from the article site. A good list of these and similar articles is provided at [19]

5.1.3 Portlet availability

One other measure of WSRP take-up is the range and availability of WSRP portlets available for consumption by prospective portals. The Portlet Open Source Trading site (POST)[13] is an open source site for organizations to share portlets developed according to the new JSR 168 and WSRP standards. This SourceForge site also includes forums for discussion. Unfortunately this site still remains somewhat underpopulated, but two popular applications are available for download there[14]:

The Google portlet is a simple portlet that searches Google using the Google search API. It comes available as a war[15] file for deployment and has been tested against Pluto. The Google API jar is also needed as well as the license key obtainable from Google. Note this example is in written in Java.

The RSS portlet is a portlet that views RSS 0.91 and 2.0 newsfeeds. It includes the edit mode for adding or eliminating additional newsfeeds.

Two alternative initiatives that act as portlet sources are the Java Portlet Community Site [11] and the file section associated with the Yahoo! mailing list [9] managed by Punit Pandey (which contained 15 portlets in February 2005).

The other sources of portlets are either portlets that are commercially available with a product, which may be standards-compliant or built to the product proprietary API, (see, for example, Knowledgeworks[16] which advertises portlets for SCORM-compliant e-learning applications, or the BEA Weblogic sample library), or occasionally code available with articles, such as ones mentioned above.

5.1.4 Take-up in HE and FE (with emphasis on the UK)

Within the UK Higher and Further education sectors JISC-funded portal-related activity has taken place, in addition to institution-led initiatives. Two prominent examples are the PORTAL [20] project which resulted in the use of the uPortal product at the University of Hull and the SPP project [21] which carried out development within the Jetspeed framework, with a subject-specific focus. Both these projects have produced substantial documentation about their activities ranging from reports of their development experiences to conference presentations. Additionally, JISC has released a number of case studies (some contributed by the above projects) regarding portal activity in UK HE and FE institutions [22]. The UK has hosted meetings as part of the uPortal JA-SIG[17] activities over recent years [23] and a UK

[13] http://portlet-opensrc.sourceforge.net/
[14] To obtain the code, click the POST link on the main page, then click [View ALL Project File] on the SourceForge site.
[15] A WAR is a Web application archive, a package of files relating to a web application that facilitates deployment to a server.
[16] http://www.techniques.org/products_knowledgeworks_portals.php
[17] Java in Administration Special Interest Group

branch of JA-SIG[18] has developed. A mailing list exists for discussing portal-related issues in the UK[19], hosted by JISCMail, and the University of Bristol maintains a list of links of portal resources and implementations, which gives an idea of the institutions participating n portal development.

5.1.4.1 Use of WSRP

Despite this level of portal activity, the scope and its timing make it difficult to gauge the indication of interest specifically in WSRP, since the period covered by the activities mostly predates WSRP. PORTAL and SPP both used Java platforms, working with versions of the portal products which were based on the JSR168 standard, with a stated timeline for moving towards WSRP development support. It would be expected that WSRP support will follow as versions of the software (uPortal and Jetspeed, or other packages), are updated within institutions. On the other hand, a move to WSRP would also need to be motivated by suitable WSRP applications becoming available which fulfil HE/FE portal needs.

During the course of this study, a small number of instances were located which demonstrate emerging awareness and experimentation with the WSRP standard. The University of Oxford is currently undertaking portal work[20] and at least one developer has been actively involved in testing WSRP support and contributing to the local (UK) and international discussion of the standard [24].

The Connect WSRP trial, on the other hand, is an initiative by JISC and the Higher Education Academy to bring together information, resources and community building opportunities in the form of portal services that can be found in one site, or individually embedded in the sites end users frequent. The Connect portal has been specifically designed as a set of discrete services which can be incorporated within an external portal or web site in order to provide functionality for users. The services are primarily designed to meet the needs of staff within universities and colleges who support teachers (including librarians, learning technologists, staff developers and curriculum developers) and include an indexed set of resources for learning and teaching and a searchable funding database. The project recently called for collaborators interested in testing these services as WSRP applications[21].

CREE[22] is taking a two-pronged approach to investigating portals and portlets. On one hand it is gathering evidence of user requirements when interacting with a range of systems and services within an institution, such as VLEs and institutional portals. The other strand of work is technical and concentrates on making search tools available through standards such as WSRP, investigating in detail, testing and documenting the practical integration of these tools with reference portal implementations.

---

[18] The JA-SIG UK mailing list http://www.jiscmail.ac.uk/lists/jasig-uk.html
[19] JISCMail PORTALS list http://www.jiscmail.ac.uk/cgi-http://www.jiscmail.ac.uk/lists/PORTALS.html
[20] http://www.oucs.ox.ac.uk/portal
[21] The CONNECT services http://www.connect.ac.uk/ixbin/hixltp?_IXSESSION_=tJ_1wVjPpOl&_IXACTION_=file&_IXFILE_=templates/welcome_embed.html
[22] The CREE project web page http://www.hull.ac.uk/esig/cree/

The GRID community in the UK is showing signs of engagement with WSRP, mainly through involvement with GridSphere, the portal product which has its roots in the Grid community. In February/March two events, a workshop and a training session[23], were hosted by the UK National eScience Centre (NeSC)[24].

The following extract from an email to the WSRP4J mailing list[25] perhaps best typifies the likely current general position of developers and strategists working within Universities:

"*We are a university implementing a student portal via uPortal and are trying to determine whether WSRP would help us meet the needs of application developers on campus who don't want to learn java to create a JSR 168 compliant portlet, but would instead like their (e.g. ASP, PHP) applications to somehow be "plugged in" to the portal. UPortal currently only has support for WSRP Consumer (they deprecated WSRP Producer), and it says future implementations of WSRP will follow the WSRP4J standard."*

Whilst the potential role for WSRP to portal-enable university applications is being assessed and is generating interest, there is still uncertainty when judging how advanced product support is in practice, and the implications of going down the WSRP route are not yet fully understood, particularly from an implementation perspective.

5.2 Implementations

A review of online sources carried out in September 2004 reveals that there are a number of open source and commercial toolkits for the development of portlets and portals, supporting JSR168 and/or WSRP. These are offered on a number of language and development platforms – the products are listed in table 1 below and more detailed summaries are found in Appendix B. The products range in sophistication from complete enterprise portal solutions to simpler libraries which provide a portlet container for developing and running portlets. A full review of all the available products is beyond the scope of this report – the table in Appendix B shows that there is a choice of commercial or free solutions. The information in the table was compiled from the product web pages. Two of the products (Pluto and Jetspeed 2) are described in detail later in section 6.2, since these products were installed and used during the practical part of this study.

[23] http://www.nesc.ac.uk/esi/events/571/ and http://www.nesc.ac.uk/esi/events/549/
[24] National eScience Centre http://www.nesc.ac.uk/
[25] WSRP4J is an Apache open source project to facilitate quick adoption of the WSRP standard by content and application providers and portal vendors. The WSRP4J users mailing list is at http://nagoya.apache.org/eyebrowse/SummarizeList?listName=wsrp4j-user@ws.apache.org

Table 1 Summary Table of Products

Product Name	**Company/ Organisation responsible**
Pluto/WSRP4J	Apache
EXo platform	EXoPlatform SARL
GridSphere	GridLab project (funded by EU under IST)
UPortal	JA-SIG
Jetspeed 2	Apache Jakarta
Liferay Enterprise Portal	Liferay
Oracle AS Portal	Oracle
Sun Java System portal Server 6	Sun
Vignette V7 Portal Services	Vignette
WebSphere Portal and Portal Toolkit	IBM
WebLogic Portal 8.1	BEA
Plumtree	Plumtree software
BowStreetPortletFactory	Bowstreet
Clickmarks PortletFactory	Clickmarks
Kapow	Kapowtech

6 Developing WSRP applications

6.1 Practical Implementation

The first part of this report was based on desk-research and provided a review of the maturity of WSRP as judged from the information available on products and support. The second part of this study takes a more pragmatic approach, by describing the production of WSRP applications from a developer's point of view. In this section the actual experiences of working with a JSR168/WSRP development platform are reported, and an example application (RDN-Include) is used to illustrate different development options.

6.1.1 Development styles for portlets

As mentioned, a portal acts as an interface with the user, aggregating content from various sources. There are various strategies that may be employed to integrate and deliver local and remote content. WSRP provides the opportunity for adopting a standardised way of delivering and re-using content from different sources, particularly remote ones. In the following sections, different models of acquiring and generating content are examined, placing WSRP in the context.

All models ultimately result in the generation of HTML content for integration into a portal, however the processes by which the HTML is generated may vary and may affect the degree of integration, portability and potential for re-use of a specific portlet.

6.1.2 The WSRP producer-consumer model

Producers and consumers are roles defined by the WSRP standard. These two roles take on different responsibilities within the interchange needed to generate, aggregate

and process interactions so that mark-up from content sources can be presented to a user.

The Producer is defined as the actor that provides a set of Web Service interfaces for the use of the WSRP Consumer. The services include *self-description* so that the Consumers can find out information about the Producer and the services (or portlets) that it offers and *Mark-up* to interact with the content fragments. Some of the interfaces that Producers could support (such as registration, which represents a relationship between the two actors) are optional.

The Consumer communicates with the Producers. It gathers and aggregates mark-up from the Producers and presents aggregated pages to its users. A typical example of a consumer is a portal.

The third role is that of the portlet. Portlets are the actual web applications that the producers offer to consumers, and they generate the mark-up that is then re-used for presentation to the end-users.

The place of GroupLog within this model is as a potential producer, i.e. it is a content-provider, wishing to develop a WSRP application for consumption by Consumers. More specifically, through the use of one or more portlets, GroupLog wishes to make a set of Web Services that can be made available remotely to third parties through a standardised way, producing Mark-up for presentation and interaction with the GroupLog application services.

6.1.3 Generating the Mark-up

The processes taking place within the portlet can be separated functionally into two kinds, those dealing with the *application logic* and those dealing with the *presentation logic*. To consider a straightforward example, for a 'simple calculator' portlet, the application logic takes in two numbers, adds them together and returns a result, and the presentation logic prepares the HTML for the display of the input and output. For example in the case of the simple calculator, the HTML could be a form where 2 figures are entered to be added, and the output is an HTML table displaying the result and the inputs, containing suitable wording etc.

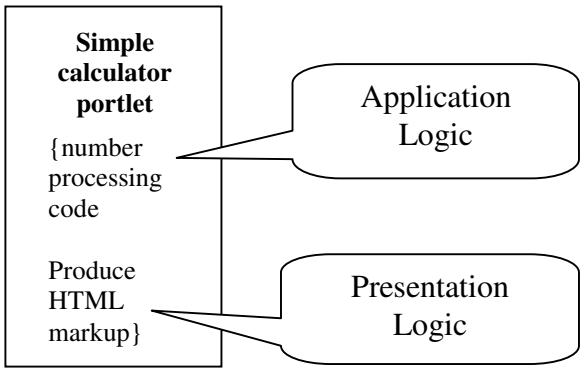

Figure 2: Portlet Application and Presentation Logic

One way to deliver a portlet is for the portlet to implement both the application processing code itself and the presentation code (i.e. produce the markup), as depicted in the above diagram.

Alternatively, the portlet could take advantage of an external application that takes care of the application processing. This could be a number of different things e.g. a CGI application, a SOAP service, an external library.

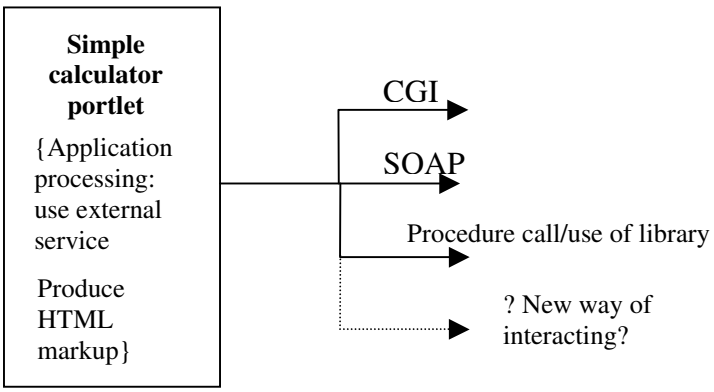

Figure 3: External services can be re-used through a number of different interfaces

The portlet then has to take care of the presentation logic, preparing it into a suitable form for display in the portal. E.g. if the externally-called application returned XML, it could be transformed into HTML by the presentation code within the portlet. The portlet needs knowledge of what the external application is returning, and has to implement the necessary presentation logic depending on its knowledge of the content that is being returned, and the output that the portlet should generate. The portlet may also have to take care of 'URL rewriting' (discussed later).

6.1.4 Local and remote portlets

A consumer that consumes a remote WSRP application is typically a portal, and will often present the WSRP-enabled portlet alongside local portlets, hosted within the local portal server. Local portlets can themselves be designed to implement application logic in the different ways as described above. The local portlet then delivers page fragments (e.g. HTML) for display in the Portal. By default, the locally-hosted portlet is of necessity developed in the same language as that of the hosting portal and is only available to be called by the local portal. If some one else wants to re-use the portlet, they have to take the code and re-install that portlet in their local portal – this might require modification depending on the portal API.

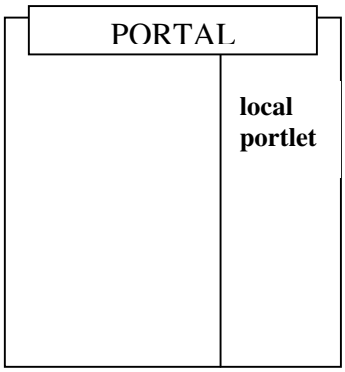

Figure 4: Model describing a portlet that is physically installed and hosted locally by the portal.

The second model, as illustrated in figure 5, adopts a Web Services model of developing portlets. The portlet still carries out some application logic and presentation logic, but the portlet can be developed in any language and used remotely by other portals. There are standard ways of calling the portlet's 'application logic' and the portlet always returns the HTML (or other markup) in response to a 'getMarkUp' request. The remote portlet is often present to the local portal in the form of a 'proxy portlet'. WSRP fits into this remote model of development.

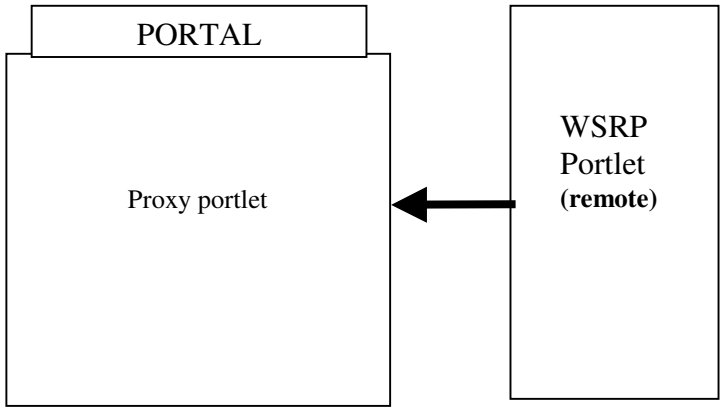

Figure 5: Model describing a portlet that is remote from the portal, using WSRP.

The way that the remote portlet generates the mark up and carries out the application logic processing can still vary as described previously. The WSRP portlet can implement the application processing itself, or it can use one or more external services to carry out the application processing.

The main difference between the two models is that, with WSRP, the application logic delivered by the portlet is hidden completely from the consuming portal and the portal can re-use the remote portlet, knowing that the mark up will be returned in a standardised way. The advantage of the second model (where the web service is WSRP) over the first model (where the portlet is local and, say, calls an external web service) is that in the second case little specific knowledge of the external web service is required by the consuming portal, and no local HTML processing is needed. Furthermore, the WSRP portlet becomes available for remote re-use by any number of portals.

When enabling legacy applications, the choice of whether to develop local or remote portlets, and what style of application logic development to use depends on a number of factors, e.g.

- If web services that are specialised to deliver the application processing already exist, it may be sensible to re-use them.
- If it is not feasible to re-develop a legacy application as internal portlet code, a way for the portlet to interact with it may be found.

6.1.5 The relationship between WSRP and JSR168

In practice, the portlets that are made available as WSRP by producers to consumers need to run within a portlet container. The Producer in WSRP acts as a 'container of Portlets'. It is useful in this context to discuss JSR168 [25], a complementary standard to WSRP.

JSR168 defines a standard Java technology-based model for portlets in portal servers that are built on a Java platform. JSR168 defines a standard Java portlet API, a portlet container, and the contract between the API and the container [17]. The set of APIs provide a uniform way for the portal container to deal with new portlets. If a portlet is written to comply with JSR-168 it should be deployable into any portal container that has JSR-168 support [26]. This role for JSR168 is distinct from the role of WSRP, which defines interoperability of remote portlets. Whereas JSR168 enables portability and vendor independence between Java products that support JSR168, WSRP further provides independence of programming languages and platforms.

One common method of implementing and consuming a WSRP application is to work within a JSR168 compliant platform. JSR168 platforms often support interaction with WSRP by means of a proxy portlet[26]. A locally-hosted JSR-168 compliant portlet can be exposed as a remote WSRP application by the portlet container. The portlet container provides a local adapter which presents the portlet to external entities as WSRP. Dually, a portal that consumes WSRP from an external source often does so by presenting it to its local JSR-168 compliant portlet container wrapped up as a JSR168 portlet.

[26] Within a portal, a proxy portlet is one that is used to stand in for a portlet, often working as an adapter to hide variations in the technology used.

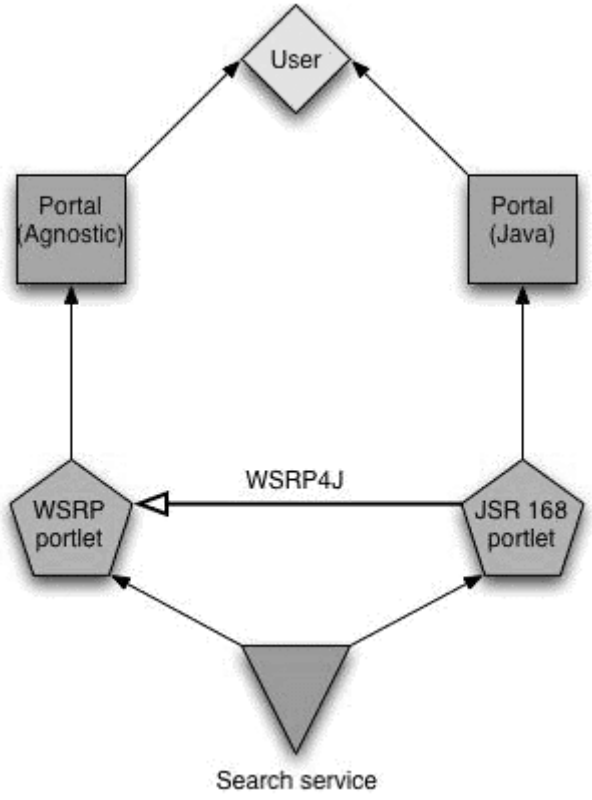

Figure 6: Development path for standards-based portlets
Re-used (with permission) from [3]

6.1.6 Does it have to be Java?

In theory it should be possible to develop WSRP in isolation of JSR168, since WSRP is specified as a set of Web Services, defined by WSDL. Although care was taken to align WSRP with JSR168 during its development, WSRP is defined as a stand-alone standard. WSRP-only development would be possible if either (i) toolkits were available for programming languages that made WSRP development as straightforward and widely supported as Web Service development is currently or (ii) WSRP-compliant interfaces were developed from scratch around an application. In practice, the first option is not available at present. The second option is not necessarily more attractive than re-using the facilities available around JSR168. The desk-research reported in the previous section of this report unearthed no evidence of toolkits for straightforward WSRP development[27]; neither could any examples of direct WSRP-enabling of legacy applications (just by using web services) be located.[28]

[27] i.e. WSRP development detached from use of JSR168

[28] These findings agree with those reported in [http://www.ariadne.ac.uk/issue41/awre-cree/#36] [own emphasis added]: "To assess fully the possibilities of using these standards to build the demonstrators it was thus decided that portlets using both standards would be developed for each tool. Initial investigations …. examined the requirements for using each standard and how this related to their existing tool(s): JAFER and HEIRPORT are Java-based already, whilst Xgrain and Balsa are written in Perl. **This initial investigation revealed independently at each partner that from a development viewpoint JSR 168 has proven easier to work with at this stage**. Factors such as the software being used to build the portlets, the requirements of the standards, and the requirements of the main testbed portal framework being used within CREE, uPortal, have contributed to this finding. **For the Java-**

On the contrary, support of WSRP closely tied to support for JSR168 is widely available. It is clear that the start of JSR-168 as an API for managing portal to portlet interaction has shaped the market. The trend within products has been to move from custom-API to JSR-168 API to added-on WSRP support. This trend can be understood in the light of historical development of interest which was first focused on delivering portals as individual silos, interacting with portlets adapted specifically to working within that portal environment, and acting as single point of entry for the end-user. Later, the motivation to provide stand-alone, remote re-usable portlets developed.

As evidenced by the product table in Appendix B, products are overwhelmingly Java-centric. This is also true for other kinds of resources. A number of available sources that address the development of portlets are directed at Java development e.g. the book on Building Portals with the Java Portlet API [27].

For those with the budget to go for a commercial product (with its attendant support), then alternatives to Java are available – for example Oracle claims to easily enable the interfacing with applications in ASP and PHP. The pre-selection of WSRP as a standard way of delivering portlets also precludes some alternative approaches that some products offer for including remote web sites into a portal (for example uPortal version 2.1 supports a web-proxy type of channel, called a CWebproxy channel[29]).

The most widely-supported option emerges as a solution built around the JSR168 API combined with added-on WSRP capabilities. This option offers the most product choice, and free, open source availability.

6.2 The RDN development Platform

In order to test the feasibility of offering GroupLog as a WSRP application, a portlet development and testing platform was installed on an RDN machine. Access to a development platform was required to enable the development team to gain first hand implementation and development experience of a WSRP software toolkit. This would enable the team to:

1. Assess the maturity of some of the available toolkits
2. Experiment with different models of WSRP development
3. Illustrate the alternative approaches with real-world examples of WSRP applications
4. Make informed decisions on the feasibility for GroupLog

---

based tools, this was a logical choice in any case. Interestingly, it also proved to be a logical choice for the Perl-based tools as there is no current toolkit to enable Perl tools to be presented as WSRP portlets."

[29] CWebProxy allows incorporation of web-based services as channels, regardless of what technology is used to implement them. It provides mechanisms for connecting to and rendering HTML and XML services. Pages are refreshed and kept in-channel when they change. HTTP standards are followed, allowing communication between the browser and dynamic back-end applications. Mechanisms are provided for passing user-specific information to the back-end application, as well as ways to support local interface technologies on a per-channel basis. (Such as encryption, shared secrets, single-sign-on, modification of http request headers, etc.)
http://www.uportal.org/developers/channel_docs/reference/webproxy/

Two platforms were evaluated in this study. These are both free, open-source options which support JSR168 and WSRP.

The first platform chosen for the RDN consists of Pluto [28] and WSRP4J [29].

Since portlets are typically presented and consumed through a portal, some form of working portal implementation was needed for testing purposes. For the purposes of this study, some aspects of portal management such as co-ordinating portlets, user profile management, personalization, single sign-on were not required as the main focus of interest is the presentation and interaction with a single instance of a portlet. Therefore some of the other features of portals were clearly out of scope and presented a potentially undesirable overhead. A fully-featured portal implementation was not considered necessary to test the portlets; at best a complete portal would be introducing additional elements which would be completely ignored in the feasibility study. At worst it would require a maintenance overhead that would not be contributing to the knowledge of interaction of the portal with the portlet. To avoid the potential overheads, Pluto was chosen, since it provides those aspects of the portal, mainly the portlet container, which are required to interact directly with the portlet, whilst doing away with the other portal-centric features (for example those that deal with the management of more than one portlet, content and layout customization, elegant configuration management tools and search capabilities) considered out of scope for the feasibility study.

From the Pluto homepage [28] *"Pluto normally serves to show how the Portlet API works and offers developers a working example platform from which they can test their portlets. However, it's cumbersome to execute and test the portlet container without a driver, in this case, the* portal. *Pluto's simple portal component is built only on the portlet container's and the JSR 168's requirements. (In contrast, the more sophisticated, open source Apache Jetspeed project concentrates on the portal itself rather than the portlet container, and considers requirements from other groups.)"*

Pluto is the reference implementation of the Java Portlet Specification [25]. It is a portlet container which manages the lifecycle and request processing of portlets which adhere to the specification. As a portlet container it provides a place for portlets to reside and nothing else. However, Pluto also comes with a minimal portal for testing portlets. Pluto's portlet container can run WSRP portlets as a consumer as well as a producer.

WSRP4J is a platform for developing and hosting WSRP compliant web services. In the definition of the WSRP standard and the JSR 168, the OASIS Technical Committee and the JSR 168 Expert group have closely collaborated to make sure that that both fit together well in portal architectures. JSR 168 compliant portlets can be exposed as WSRP compliant web services and conversely, WSRP services can be integrated through generic portlet proxies written to the Portlet API. WSRP4J provides both consumer and producer modules, which can be installed separately and have different requirements. The installation instructions describe how to install the two components within Tomcat (note Tomcat 4.1.24 or higher, JDK 1.3.x are prerequisites).

An initial implementation of Pluto was deployed on an RDN machine by downloading the source and installing it according to the instructions[30]. The Pluto installation required Java (standard edition version 1.4.1_02 was already installed), Maven[31] (version 1.0 was installed) and Tomcat (version 4.1.18 was already installed). The implementation of the Pluto Portal and Portlet container on the RDN platform is located at http://walrus.rdn.ac.uk:1976/pluto/portal

Despite following the instructions on the Apache website, the first installation of Pluto was only partially successful. The Pluto Portal Driver (i.e. that component of Pluto which is the minimal portal available for testing portlets) was displayed (Fig. 7), however attempts to call the test portlets that are deployed with the installation resulted in errors being reported (Fig. 8). The reasons for this failure remained obscure.

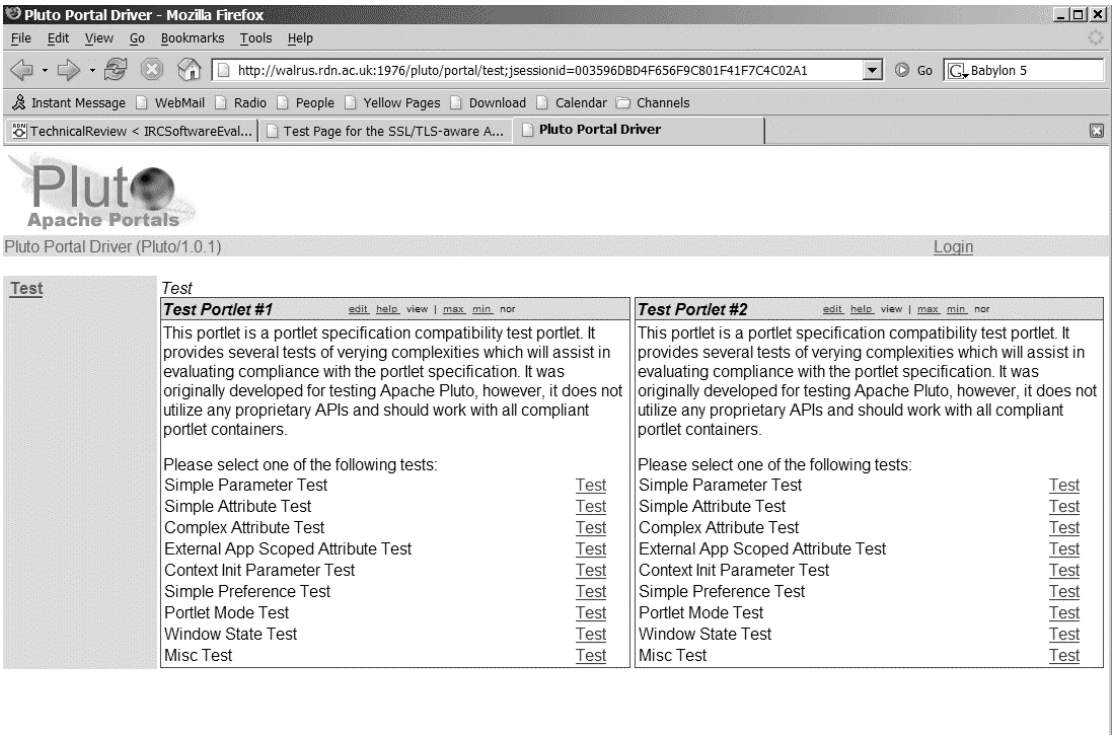

Figure 7: The Pluto Portal driver installation showing the test portlets

The installation of Pluto (version 1.0.1-rc1) was subsequently reattempted on an alternative machine that became available running updated versions of Java (version 1.4.2_04) and Tomcat (version 4.1.30). This time Maven version (1.0.1) was used (a newer release that had by then become available). This time the test portlets were deployed and displayed successfully (Fig. 9).

[30] http://portals.apache.org/pluto/install.html#Installing_Distributions

[31] Maven is a software project management and comprehension tool which can manage a project's build, reporting and documentation from a central piece of information. http://maven.apache.org/start/download.html

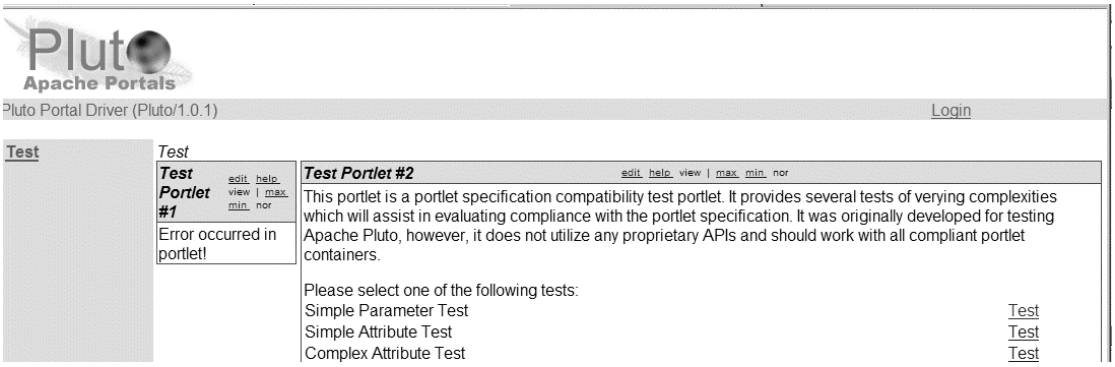

Figure 8: Unsuccessful installation results in errors on the test portlets page

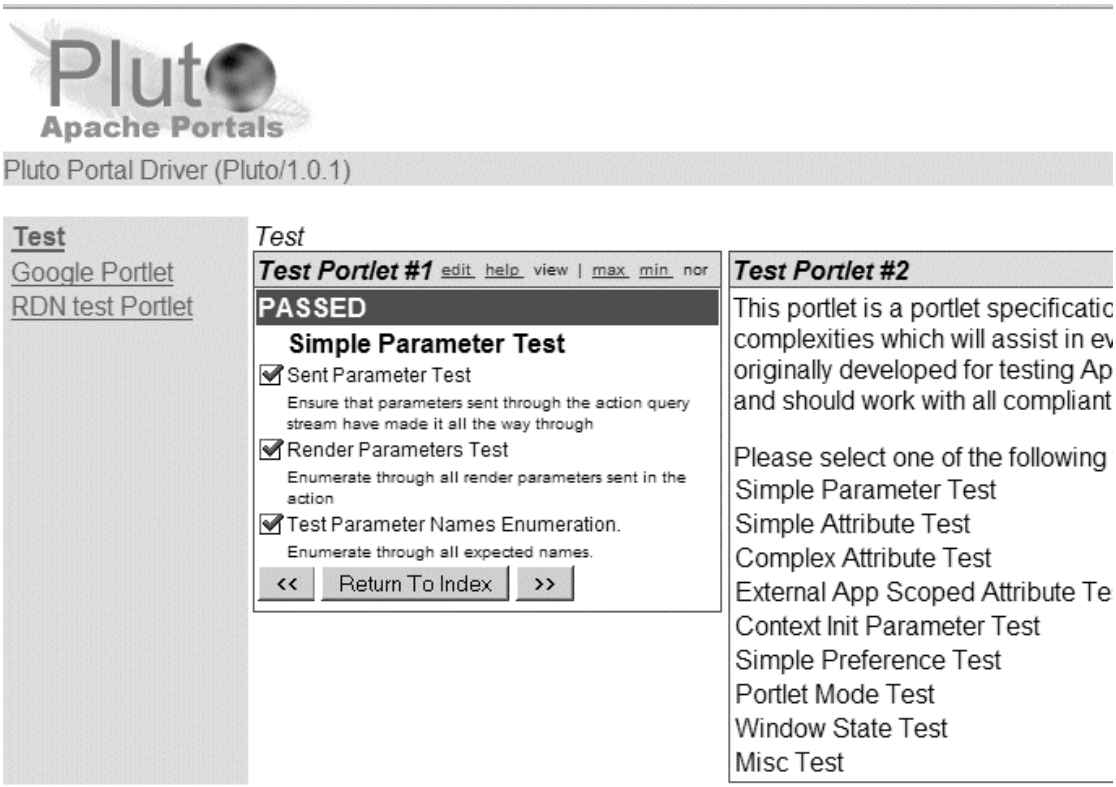

Figure 9: A successful Pluto installation was achieved

A first version of an RDN portlet which displays a search box for entering user queries was deployed. However, a problem was encountered when attempting to pass search results back to the test portal for display. Extensive searching of the Pluto mailing list unearthed some messages[32] which indicated that there was incomplete feature support in the Pluto demo portal implementation. Note this is not a problem of the Pluto portlet container, which is the part of the API which communicates with the portlet code, but of the driver required to view the portlet and test it is functioning within a web browser. The specific problem encountered is related to the ability to

[32] http://nagoya.apache.org/eyebrowse/SearchList?listId=&listName=pluto-user%40portals.apache.org&searchText=setRenderParameter&defaultField=body&S and http://nagoya.apache.org/eyebrowse/ReadMsg?listName=pluto-user@portals.apache.org&msgId=1720124

control rendering parameters i.e. the passing of information within the portlet that relates to values intended for rendering as output of the portlet.

At this stage, it was decided that a change of strategy was required. The use of rendering parameters was considered necessary to the progress of the feasibility study and an alternative platform for portlet development was chosen.

Jetspeed 2

Jetspeed 2 [30] is an Apache project and is the successor of Jetspeed 1. It is an Open Source implementation of an Enterprise Information Portal, using Java and XML. The goal is to make Jetspeed a tool for both portal developers as well as user interface designers. Currently the focus is on providing developers with a set of tools that facilitate building the base for the portal. With Jetspeed you can quickly build an XML portal and also syndicate your own content. Jetspeed has the following requirements: Ant 1.5 or higher, Maven 1.0 or higher, Java 1.4.2_02 or higher, Servlet 2.3: Tomcat 4.1.x or Tomcat 5.0.28 or higher.

For the RDN installation, Jetspeed 2 (beta version) was deployed into Tomcat (version 4.1.30).

6.2.1 RDN Include: an example application for assessing implementation styles

Due to the tight timescales for the GroupLog project it was decided to choose an existing stable application which could be redeveloped as a portlet; the chosen application had the benefit of familiarity for the developer, and therefore there would be no learning curve; it also provided an unchanging application whilst the GroupLog is in re-development. The intention was that the RDN-I application would provide a good yardstick against which to compare the options for GroupLog, and provide feedback in the early stages when the GroupLog re-development could be shaped and influenced. A parallel interest within the RDN in exploring portal solutions also proved convenient since the development effort could be shared with the RDN. This contribution turned out to be necessary for the success of the study since the time required for preparing the set-up and practical development for exploring the examples far exceeded the time projected for development in the GroupLog project.

6.2.1.1 About RDN-Include

RDN-Include (RDNI)[33] is a service offered by the Resource Discovery Network (RDN) as a means to include the services of the RDN into external web sites. One of its functions is to allow searches using the RDN ResourceFinder facility. Using RDNI, the ResourceFinder search engine is made available to users within an institution's website. A search box is displayed in which users enter search terms. The results of the search term are returned within the institutional web environment, retaining the web site's look and feel. The user does not have to leave the institutional domain. An example of RDNI can be seen at De Montfort Library website:
http://www.library.dmu.ac.uk/cgi-bin/RDN/include.cgi/

[33] http://www.rdn.ac.uk/rdn-i/

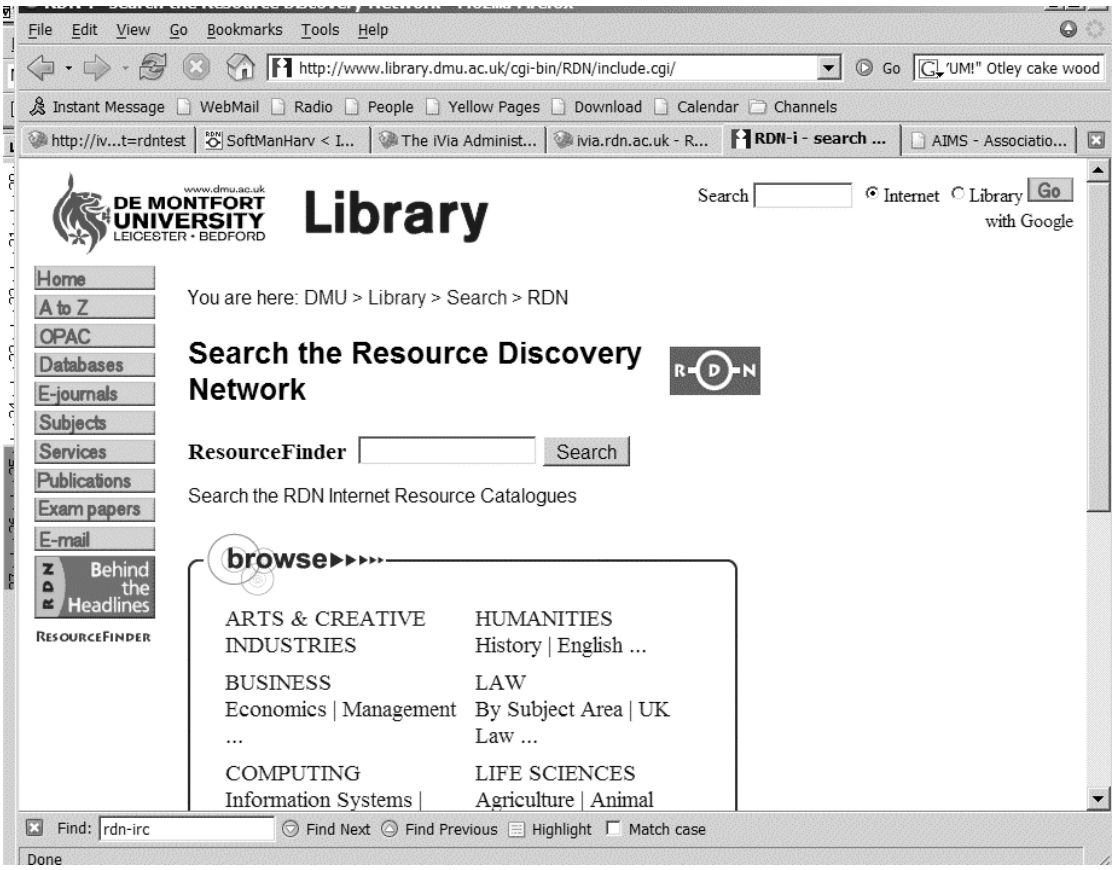

Figure 10: The De Montfort University Library installation of RDN Include

6.2.1.2 Technology behind RDNI.

One of the technologies behind RDNI is a CGI script written in Perl[34]. The script receives RDN-I requests via HTTP, uses ResourceFinder to generate a set of results, formats those results in preparation for display at the end-user site, and forwards the resultant HTML to the original site making the requests. RDNI uses what is called a 'tag' facility to allow for the construction of relative links so that the search results can be browsed within the calling application without leaving that website.

[34] Other forms are available, such as a solution using Javascript, which are not referred to further here.

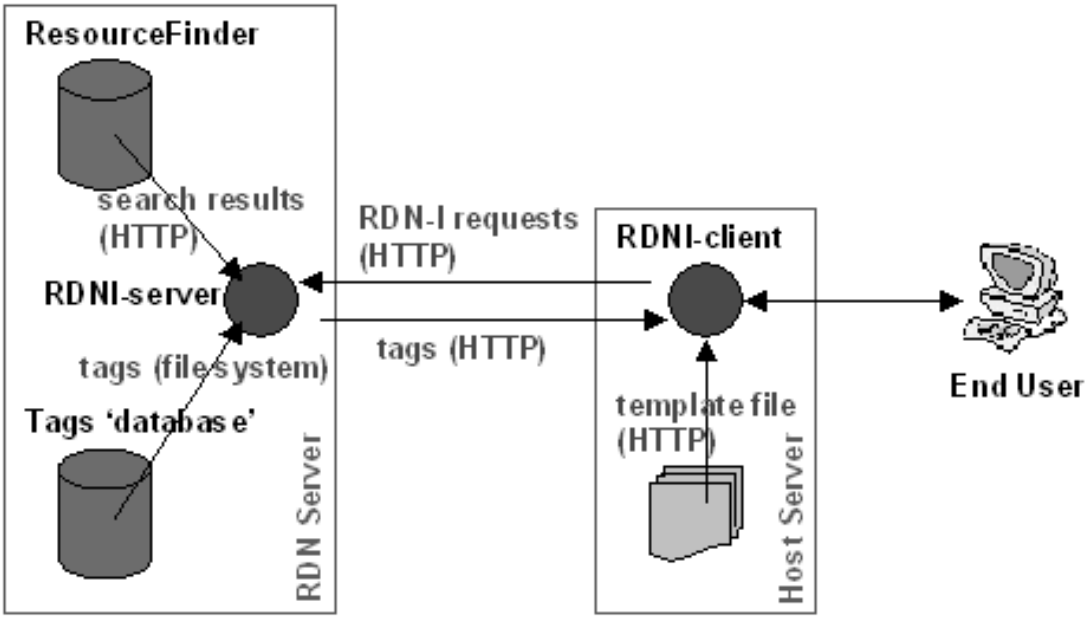

Figure 11: RDN-I uses HTTP requests and a tag facility to deliver content to clients

An example of the script in action can be seen by using the URL:
`http://www.rdn.ac.uk/rdn-i/cgi-bin/rdnisearch.cgi?query=[searchterm]`
where [searchterm] is the user query, e.g.
`http://www.rdn.ac.uk/rdn-i/cgi-bin/rdnisearch.cgi?query=Aristotle`
The script takes some other parameters, such as

- 'Start' which indicates the position of the result in the result set
- 'Set' which allows limiting the search to certain subject areas

6.2.1.3 Accessing the RDN through a web services interface

An alternative machine to machine (m2m) method to access the RDN ResourceFinder is available by means of the SRW[35] protocol. The RDN provides an experimental implementation of SRW[36] which supports search and retrieve requests conforming to the protocol. An accessible introduction to SRW is available at [31]. In a nutshell, SRW defines a Web Services operation *searchRetrieve* (described by WSDL) in which the client sends a *searchRetrieveRequest*. Besides the search query (i.e. the search term) the request accommodates parameters such as the maximum records to return in the response, and the XML Schema that the records should be returned in (the latter obviously depends on the schemas that the service supports – the RDN supports Dublin Core). The response consists primarily of a list of XML records that matched the search, along with the full count of how many records were matched.

Previous RDN work with SRW resulted in the generation of Perl and Java clients for accessing the SRW interface. These clients are SOAP clients that use the available WSDL to makes calls to ResourceFinder and return RDN search results. The Java clients were automatically generated from the WSDL for the SRW service, using the WSDL2Java facility. The Java client is an Axis[37] 1.1 program. In the process of

[35] SRW is the Search and Retrieve web service, a web service for searching databases containing metadata and objects http://www.loc.gov/z3950/agency/zing/srw/
[36] http://www.rdn.ac.uk/publications/workingwithrdn/
[37] Axis http://ws.apache.org/axis/

generating the client, a number of Java classes corresponding to the concepts of SRW are created. These classes are used within the SRW client when interacting with the SRW service, and support the manipulation of the returned results, which are presented as records. The iteration through the records and extraction of fields from the records depends in turn on the use of other standard Java classes for XML-processing [see package org.w3c.dom http://java.sun.com/j2se/1.4.2/docs/api/org/w3c/dom/package-summary.html].

6.2.1.4 Development of RDN-Include as a WSRP application

The two machine interfaces outlined above (CGI and SRW) provided two development options for exploration of delivery of a WSRP portlet version of ResourceFinder. In both cases, a JSR168 portlet was developed to handle search requests entered by users through a portal, translate this into a request to either the CGI interface or the SRW interface, and pass on the results to the calling portal. These two options were considered ideal as a case study since they mirrored the current and future status of GroupLog. Version 1 of GroupLog consists of CGI scripts which handle interaction with users through a browser. The development version of GroupLog is improving the functionality of version 1, whilst considering replacing at least some of the functionality with Web Services. CGI-based RDNI is presentation oriented since it was intended for producing output for display in Web browsers and integration into Web pages. On the other hand, the SRW interface separates application logic from the presentation; it is intended for m2m use, and its output is a machine-oriented XML-encoded result list. The presentation is left to the calling application to deal with.

6.2.1.5 Overview of Portal-Portlet interactions using JSR168

To recap, a portal is an application which aggregates portlet applications together in a presentable format, supporting facilities such as user customisation and single sign-on mechanisms. A portlet is an individual web component that is made accessible to users via a portal interface. Users issue requests against portlets from a portal page. [5]

A portlet container sits between a portal and its portlets. It provides the run-time environment to portlets, and manages portlets by invoking their life cycle methods.

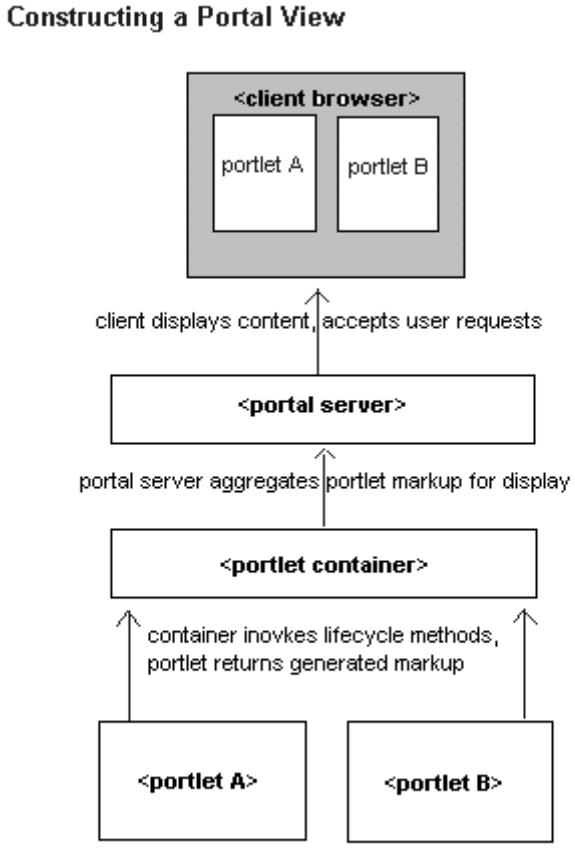

Figure 12: The architecture of portals and portlets
Re-used from [5]

The portlet is a component written in Java against the portlet specification JSR168. The specification defines the contract between portlet and portlet container, and a set of portlet APIs. The portlet takes a request from the container and returns a response; in other words, it processes requests and generates dynamic content. The portlet container manages *life cycle events* during interaction with the portlet. The basic portlet life cycle of a JSR168 portlet is (1) Init: initialise the portlet and put the portlet into service (2) handle requests: process different kinds of action- and render- requests and finally, (3) Destroy: put portlet out of service (collect garbage and free up portlet resources).

The portlet provides implementations of specific methods required to fulfil the obligations that must be satisfied for the portlet to interact with the container. It is beyond the scope of this study to explain in detail the JSR168 specification and the related portlet APIs and the reader is referred to the references [particularly 5, 13, 14, 15, 25] for further explanations. However, below some salient points are introduced and some code samples are provided by way of illustration of the programming involved.

Generally, the portlet implements the Portlet interface. Commonly, this is done indirectly by extending a generic portlet class that has already implemented the Portlet interface. The generic portlet class is a convenience class that defines three empty

methods, doView, doEdit and doHelp. The portlet for a specific application extends the generic portlet class and over-rides these specific methods that the interface provides.

The render() methods

- are requests that display the application in its current state at any given point
- are invoked when the processAction() method has completed
- can be invoked when a user triggers a render URL in the interface
- produce mark up depending on the state of the portlet

processAction() methods

- are invoked by a user clicking an action URL in the portal interface
- are usually requests that command the portlet to change the state of the underlying application
- one action per client request is triggered

Note: Some aspects of portlet management (such as Portlet Mode and Window State) are omitted here since they are not necessary for explaining the illustrating code.

6.2.1.6 Development of a CGI-based portlet

The CGI-based RDN-Include portlet uses a JSP[38] to display a search form to the user, then takes the input (a search term) from the user and connects to the RDN-Include CGI script by constructing the appropriate URL (containing the user's search term) and making an http request. It then processes the output from the CGI request (basically results for the search, marked up as HTML) and passes it on to the portal for display using the JSP. The user can then make another query.

The first time the portlet is called, the doView() method is called. This sets the mime type and sets up the search form for the user to enter the search term, making use of a JSP.

```
public void doView (RenderRequest request, RenderResponse response)
throws PortletException, IOException
    {
        response.setContentType("text/html");
        String jspName = getPortletConfig().getInitParameter(
"jspView");
```

Portlets can include a JSP page to render the output; the mechanism used in JSR168 is based on that for servlets and JSP pages in the servlet API.

The JSP contains the HTML for the web form:

```
<form action="<%=doSearch%>" method="post">
<input type="text" name="query" class="textinput" value="" size="15">
<input type="image" src="http://www.rdn.ac.uk/images/go.gif"
width="34" height="32" name="Submit Search" border="0" alt="Search
over 100,000 descriptions of high-quality Internet resources relevant
to the higher and further education sectors.">
</form>
```

[38] Java Server Page (JSP) a technology based on Java to develop dynamic web pages; JSP files are HTML files with special Tags containing Java source code that provide the dynamic content. see http://www.visualbuilder.com/jsp/tutorial/default.asp for a tutorial

`doSearch` is a variable (or placeholder) within the JSP which will be replaced by a URL. The interaction of the user with the search form (entering a search term and clicking the search button) is expected to result in a processAction() method being triggered. The processAction() method is the destination within the portlet to which the values from the webform should be passed. In order to specify processAction as the destination, an actionURL is created in the doView method of the portlet.

```
   // Create the action URL that triggers RDN search
        PortletURL doSearchURL = response.createActionURL();
        doSearchURL.setParameter("search","search");
        request.setAttribute("doSearch", doSearchURL.toString(
));
```

The actionURL is created by calling createActionURL(). A parameter is added to the URL so that when it is triggered and processAction() is called, we can check that a search request has been made by checking for the presence of the search parameter.

To dispatch the content to the JSP, a request dispatcher is first retrieved. The include() method is then called on the request-dispatcher object.

```
PortletRequestDispatcher rd =
getPortletContext().getRequestDispatcher(jspName);
        rd.include(request, response);
    }
```

Once a user enters a search term and clicks the search button, the processAction method is invoked. Two objects are passed to processAction() when it is called by the portlet container: ActionRequest and ActionResponse. The parameters of the request are accessed through the ActionRequest object. In this case two parameters are available: the `search` parameter simply indicates that the request comes from the search web form. The `query` parameter contains the user's search term.

```
public void processAction (ActionRequest request,
                           ActionResponse actionResponse)
 throws PortletException, java.io.IOException
    {
        // detect that the RDN search URL has been clicked
        String search = request.getParameter("search");
        if (search!=null) // search
        {
            // get the query term
            String queryString = request.getParameter("query");
```

The next step is to call the RDN CGI script with the query term. This is done by constructing the appropriate URL, opening a connection, and reading the output.

```
            String queryURL = "http://www.rdn.ac.uk/rdn-i/cgi-
            bin/rdnisearch.cgi?query="+queryString;
            URL rdn = new URL (queryURL);
            BufferedReader rdnin = new BufferedReader(
                                   new InputStreamReader(
                                             rdn.openStream()));
            String inputLine;
            String myResult = "";
```

```
while ( (inputLine = rdnin.readLine() ) != null) {
      myResult = myResult+inputLine;
}
rdnin.close();
```

The results, stored in myResult, must be made explicitly available to the doView() method so that they can be displayed in the JSP. This is achieved using setRenderParameter()

```
actionResponse.setRenderParameter("result", myResult);
```

Additionally, to make the user's initial input values available in the render method or the JSP, the actionResponse.setRenderParameter() method is again used. This makes the search term entered by the user available to the render() method to display it again to the user "You searched for <search term>"

```
// Send query term to render
actionResponse.setRenderParameter("query", queryString);
```

Once processAction completes, the doView method is executed again. The search term and the response can be accessed through the renderRequest object, since the parameters were set during processAction. They are then made available to the JSP using the setAttribute method.

```
request.setAttribute("rdnresponse", request.getParameter("result"));
request.setAttribute("query", request.getParameter("query"));
```

Within the JSP, the search term and the results can be displayed using:

```
<%
  String query = (String) request.getAttribute("query");
%>
<%
  String results = (String) request.getAttribute("rdnresponse");
%>
Search term is <%=query%><br />
Result is: <%=results%> <br />
```

Figure 13: The RDN results are displayed using a portlet within a Jetspeed Portal

6.2.1.7 Using Web Services

The RDNSRWportlet.java portlet is similar to the first RDNI portlet. It re-uses the search form for the input of the user's query. However this time a Java (Axis) client is used to make the web service call to the RDN ResourceFinder SRW interface. The search results are returned as XML. This provides more flexibility in display options of the search results, however it carries the overhead of processing the XML and reformatting it into HTML for presentation to the user in a web browser (i.e. the portal).

Normally, for non-trivial portlets the HTML would not be included directly within the portlet Java code. Use would be made of some presentation layer technology, such as JSP pages. The JSP would shoulder more of the responsibility for rendering the response, for example by directly processing the results. This would represent a cleaner separation of concerns. However for simplicity RDNSRWportlet.java processes the XML and formats it into an HTML table for display. The look is kept similar to the CGI version of the portlet.

The doView() method and the jsp are identical. ProcessAction() differs in two ways:

- A doQuery method is defined which handles the SRW interaction
- A convertToHtml method deals with the processing of the XML and transformation to HTML.

```
// do query and send results to render
rdnsrw.RecordsType myXMLResult = doQuery(queryString);
```

```
String myResult = convertToHtml(myXMLResult);
actionResponse.setRenderParameter("result", myResult);
```

The doQuery method re-uses the Axis web services client class that connects to the SRW service via the WSDL, and requires the Java classes generated to deal with an SRW response (as mentioned earlier)[39]. ConvertToHtml makes use of some standard Java classes for processing XML.

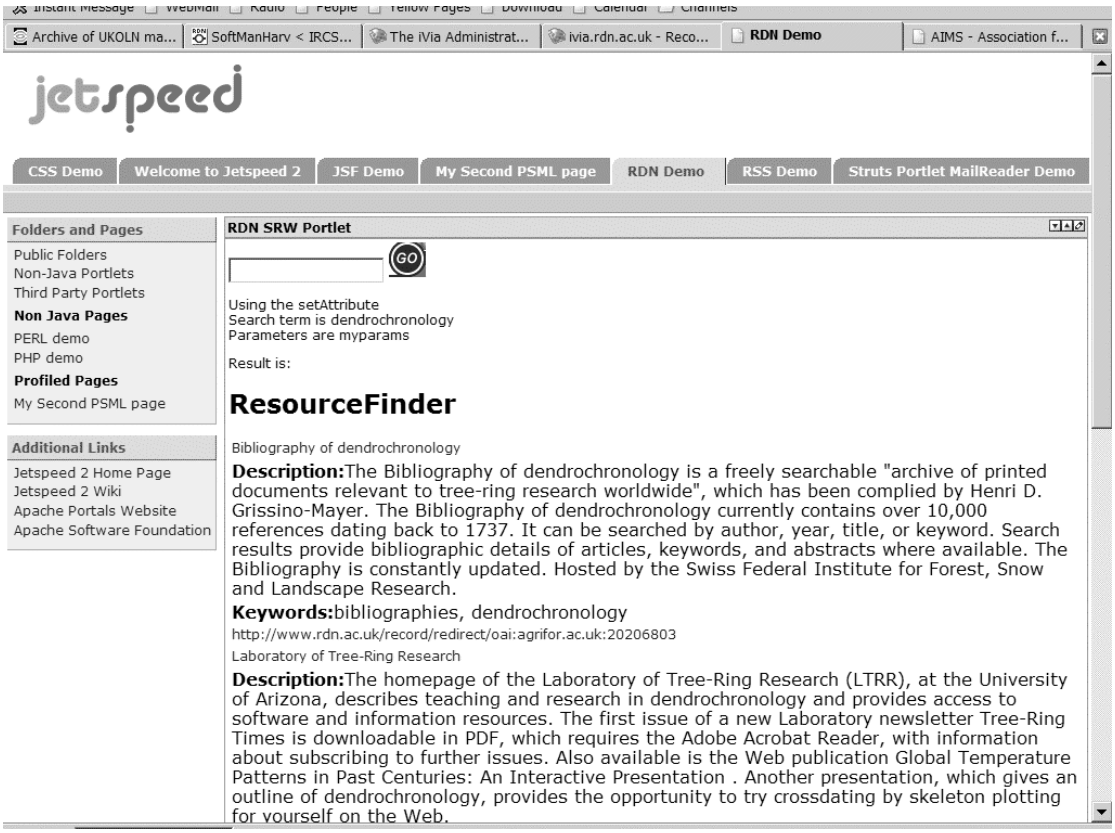

Figure 14: The SRW portlet uses web services mechanisms at the back-end, then displays the RDN results in a similar manner to the previous portlet

6.2.1.8 Deployment of a portlet into a portal framework.

The portlet application, when developed to the JSR168 standard, is a standard web application. This means that it consists not only of the portlet classes (i.e. the Java code which does the processing), but also of additional files, such as xml files which define the portlet deployment description. The files are packaged together and make up a Web application archive (also called a WAR), and are organised in a directory structure so that they conform to the accepted conventions. Although this report has concentrated on describing the coding of a portlet, the use and testing of portlets within a JSR 168 framework requires confidence with packaging and deployment,

[39] A technical issue (related to the loading of Java classes) was encountered when combining the use of Axis classes (required to make the SRW calls) in the Jetspeed framework within the Tomcat environment. A relevant entry in the Tomcat bugzilla bug reporting system was traced http://issues.apache.org/bugzilla/show_bug.cgi?id=3888
Unfortunately there seems to be no conclusive agreement as to whether this is an actual bug or if there is a definite solution that can be applied on the user's part to avoid the classloading problems. The discussion in the bug report reflects the detailed technical issues that need to be understood and grappled with (even within the development of this simple example).

and understanding of the formats of these additional XML descriptor files such as portlet.xml and web.xml.

6.2.1.9 Caveats

As mentioned, the above code snippets and the developed code were intended to explore two different methods of exposing a legacy application as a WSRP portlet. Several features of portlet development have been omitted, including the use of portlet state, preferences, portlet mode, portlet security, portlet sessions, validation and exception handling. Furthermore, the full functionality of RDNI has not been used; the ability to page through results and request different size or format of result sets has not been demonstrated.

It should be noted that the GroupLog functionality (addressed below) is more extensive than RDNI:

- GroupLog has a much larger number of use cases
- GroupLog requires user authentication
- GroupLog handles more complex interactions, such as submission and retrieval of documents

6.2.2 Conclusions from development of RDN-Include

Although the sample application is not a fully-fledged WSRP-enabled RDNI, it has shown the potential for two routes of development when enabling a legacy application, and demonstrated some of the implications.

7 Assessing the feasibility of using WSRP for the GroupLog Project

7.1 Overview of GroupLog

GroupLog is an interactive web-based teaching and learning tool. It supports collaborative activity through structured group work. For tutors it provides the logistical support for preparing and disseminating activities, aggregating responses and disseminating feedback. Through a series of web-based forms, the tutor sets up a cohort split into a number of groups, to which the students in the cohort are assigned. An activity is then defined by the tutor and the activity is allocated to one or more groups. The tutor is allowed to set parameters for when contributions for an activity should be submitted and when student's responses are published for viewing by a cohort of students.

GroupLog supports the contributions of many authors, and students can both contribute to and benefit from a knowledge pool. Groups submit their response for review by the tutor through the GroupLog website.

The main user roles of GroupLog consist of *Group Members*, who can create, view and review responses to activities, and *Tutors*, who have authorisation to create Groups and author and assign activities to Groups, and publish responses.

GroupLog was developed and tested as a prototype application by the CDNTL at the University of Bath. It is in the process of being redeveloped to include enhanced functionality and improve documentation. The intention is to add improved user

management and editing facilities. The development is being carried out within the context of the ELF. GroupLog is developed as a series of PHP scripts, using MySQL at the backend.

7.2 Implementation options for GroupLog

Based on the desk research presented in previous sections and the development experience gained on the RDN platform, the implementation options for delivering GroupLog as a WSRP application can be summarised as:

1. Redevelop the GroupLog functions in Java for close integration with the portlet code.
2. Re-use the GroupLog functionality from within the portlet through interaction with the existing (or modified) GroupLog CGI scripts.
3. Re-develop the GroupLog functionality as Web Services, and access GroupLog functionality from the portlet by using the Web Services interface.

These three options are now considered on their individual merits:

Option 1: Redevelop the GroupLog functions in Java for close integration with the portlet code

This option would imply the redevelopment of functionality within GroupLog as Java classes and libraries, including the application logic, interface generation and database connectivity. This represents a complete shift in the development plans for GroupLog entailing a change of platform, with possibly the use of JSP or a complete Java environment (e.g. J2EE).

This option can be deemed infeasible in the short-term since:

- It does not fit with current plans for GroupLog (which focuses on enhancements to the code and documentation rather than re-development; the current code-base is not Java-based).
- The expense of the 'start-up' costs needed to change development environments does not outweigh the benefits. The allocated time for the project did not include the effort needed to make this investment (i.e. acquiring the required expertise in Java), since this requirement was not known when the project was planned.

This option might have been, or could become, the preferred option for GroupLog if:

- The development team was already using Java
- Better separation between the interface functions and the transactions develops, for example if GroupLog was already Web-Service based;
- There were other motivating factors for the GroupLog team to move to a completely Java-based solution e.g. Java was chosen as the preferred Web Services development platform.

Option 2: Re-use the GroupLog functionality through interaction with the existing (or modified) CGI scripts

This approach would work along the lines of the first RDN-Include example shown. However RDN Include had been designed for the purpose of preparing the response for display in a third-party website, and supports one main type of interaction (search and view results). GroupLog was intended as a stand-alone application, uses a number of different user interfaces and supports user transactions such as authentication (i.e. logging in),
If GroupLog were to take this route for WSRP development, the implications (as derived from experience of RDN include) are perceived to be:

- A number of JSP pages would have to be developed to support the different interfaces that GroupLog presents (e.g. creating and modifying activities, viewing of activities, creating/editing of responses, creating group members and groups, assigning groups to activities, publishing responses)
- The portlet code would create the appropriate actionURLs. The parameters in the URLs would be used to select which functions of the CGI script should be called upon to fulfil the user requests. The portlet code would need to include mechanisms to detect the appropriate functions needed and construct the CGI URLs accordingly.
- The CGI scripts would have to be designed such that the different functions could be easily called; this would require a certain modularity in the scripts. Ideally, the scripts would reflect the use cases documented by GroupLog.
- The reponse from the CGI would have to be in a format that is developed in relation to the JSPs. A decision would have to be made on the balance of preparation of the HTML within the CGI scripts (it is assumed that currently all the HTML is prepared within the scripts) and the extent to which the JSP pages were used to contain the HTML.
- The responsibility for user authentication and user profiling *could* be re-assigned to the portal, i.e. the CGI would not deal directly with the user logging functions and the determination of authorisation.
- Issues of threading may need to be considered to deal appropriately with user sessions.

Option 3: Re-develop the GroupLog functionality as Web Services and access GroupLog functionality from the portlet by using the Web Services interface

In this approach, it would be assumed that the functionality (or application logic) of Group Log was available as web service calls. These would separate the actual transactions (such as storing and retrieving activities and responses) from the user interfaces required to interact with the user. The portlet would then deal with building the user interface (for example by using JSPs), whilst calling on the web services to fulfil the transactions. The advantages of this approach are that it fits in with the Web Services/Service Oriented Architecture design advocated by the ELF. However it requires knowledge of which functionality of GroupLog is best developed as Web Services, and a commitment to that solution. At this stage, it has not proven easy to determine how to repackage the GroupLog functionality into Web Services, mainly because the potential for re-usability of the services, and the level of granularity at which they would be used, is still an unknown factor. Dually, the availability of external Web Services that could replace some of the GroupLog functionality has been found to be poor.

7.3 A GroupLog example

One of the GroupLog use cases documented in the GroupLog documentation can be used as an example. The GroupLog use cases are an initial step towards breaking down the functionality of GroupLog into a modular design and documenting how GroupLog works. The use cases provide a user view of the system, outlining the user interactions with various parts of GroupLog to achieve the user's goal. A more detailed UML system design in terms of interaction and state diagrams is not available at the current time. The description of backend functionality of GroupLog in the example use case is based on discussion with the GroupLog developers and demonstration of the GroupLog prototype system.

The **viewActivities** use case allows a GroupMember to view a list of Activities, and select one of the Activities for display. The viewActivites use case is documented as follows:

1. Within GroupLog main page
2. GroupMember selects Activity from Activity drop-down menu
3. Activity is displayed in Activity area on GroupLog main page

The use case assumes the preconditions that a GroupMember is logged in and can view only the activities for which the GroupMember is authorised. The user interface for this use case is simple and consists of (1) a main GroupLog page which displays a selectable list of activities, and (2) a display of activity details within an activity area once an activity has been selected. The backend functions can be described as `DisplayActivitiesForGroupMember` and `DisplayActivity`.

`DisplayActivitiesForGroupMember` would be expected to take as parameter a groupMember ID and return a list of activities. This process would entail some checking of group membership and association with activities, with retrieval from a storage system such as a database or directory, however for the purposes of the following discussion the process can be treated as a black box that takes inputs and returns outputs. `DisplayActivity` takes as input an Activity ID and returns the details of that activity (once again ignoring the actual processes required to retrieve the activity details from storage).

In the scenario where GroupLog provides the DisplayActivitiesForGroupMember through a CGI script, and assuming the output (a list of activities) is returned as an HTML fragment which would fit into the main page interface as a drop-down list, portlet functionality would be achieved as follows:

The JSP for the main page would be constructed so as to contain the activity drop-down list, with provision to include the appropriate actionURLs (created by the portlet).
The GroupMember ID would be controlled and stored either through the portal framework support or explicitly through some other mechanism.
The portlet would be responsible for calling the CGI which makes available `DisplayActivitiesForGroupMember` and retrieving the list of activities.
The portlet would create action URLs for the drop-down list, with parameters that allow the portlet to detect that an activity detail request is being made, and containing the activity ID.

Assuming `DisplayActivity` can similarly be called within the CGI script, and returns an activity as a simple text description, or perhaps HTML table fragment, once a user has selected an activity to view by triggering an action URL:

The portlet would contain controls to detect that a `DisplayActivity` request was being made.
The activity ID would be retrieved by the portlet and used to make the appropriate connection to the CGI.
The results returned by the CGI would be integrated by the portlet into the required JSP for display in the portal.
The portlet would create appropriate action or render URLs that enable the user to move on to the next use case, for example createGroupResponse for an activity.

Appropriate tracking of user sessions, preferences and authorisation would have to be built into the application, for example through threading and use of advanced portal API features.

7.4 Testing WSRP applications

WSRP (or JSR168) portlets must be tested within a portal framework. This testing is required during development and therefore the development of portlets would usually require the support of a suitable framework within which to test. During development this could be provided locally by installing and running a compliant framework.

A further phase of testing would usually consist of the recruitment of third parties who are willing to deploy the portlet within their framework, or consume one available as WSRP. This would provide confirmation of portlet compliance; portlets built within a framework that supports standards should be portable between any conformant third party system.

Within UK HE/FE, the JISCmail PORTALS list could be used to recruit suitable collaborators. Alternatively, the institutions which have figured prominently in JISC development programmes are usually willing to assist in testing, by mutual agreement.

7.5 Feasibility Conclusions

Taking into account
- time available for development
- GroupLog application as it currently stands
- plans for the development of GroupLog (as described in the JISC proposal)
- implementation options available and the state of the art of WSRP development

Three strategies can be identified for GroupLog:
sit and wait: the GroupLog team may decide that the technology is too immature and the investment too great to make; GroupLog may wish to wait until the demand for portlets within HE grows before committing to a specific technology.
experimentation coupled with move to Web Services: if GroupLog continues with the web services development, one or two of the use cases built around the Web Services could be developed as portlets. This would give GroupLog direct experience

and understanding of the implications of portlet development and give it a head start, should it wish to undertake further work in this area.
early commitment: a third option for GroupLog would be to become an early experimenter, and act as a driver for take-up of WSRP by providing a sample application which has already generated interest in the user community.

8 General Conclusions

- The industry seems confident in WSRP; there are a number of portal vendors supporting the standard and involved in its development and promotion.
- In UK HE there is significant portal activity, particularly JISC-funded. Use of WSRP is however in an experimentation stage and happens across programmes. Current portlet development is focussed on search functions (for example the Connect and CREE services). Support of groupwork would provide an interesting alternative test case.
- There is a choice of tool support for development, however solutions are at this time very Java-centred; resources for developers tend to be product-focussed. Tools such as wizards are available mainly in commercial products. Development is not as yet a plug-and-play solution, but requires programming and technical effort.
- The main viable solution is through JSR168, requiring a Java platform with attendant support (Apache, Tomcat and development environments e.g. Ant, Maven, Eclipse). Advanced Java knowledge is required for set-up and maintenance. This approach is best suited to establishments already committed to Java or prepared to make that investment.

9 Acknowledgments

The parallel investigation being carried out by the RDN on the use portal frameworks for web resource discovery has contributed to the development and technical support of the practical implementation work that underlies the second part of this report.

10 Appendix A: Glossary

Portal	A layer which aggregates, integrates, personalises and presents information, transactions and applications to the user according to their role and preferences. [18] Technically, a portal is a network service that brings together content from diverse distributed resources using technologies such as cross searching, harvesting, and alerting, and collates this into an amalgamated form for presentation to the user. This presentation is usually via a web browser, though other means are also possible. [2] A portal can also provide a convenient single sign-on mechanism for users. [5]
Portlet	Distinct building blocks of functionality, e.g., cross-search, alerting, listing, each one offering a visible component to the user. Each building block is known as a portlet. These can be joined together to create a portal environment, within which various degrees of personalisation can be incorporated, or embedded. Portlets feature heavily in many of the current portal building frameworks such as the Apache Jetspeed project, IBM's WebSphere Portal Server and Oracle's Application Server Portal. [2] Portlets are user-facing, interactive web application components rendering markup fragments to be aggregated and displayed by the portal. For example, a weather portlet that could be displayed with a stock quote portlet on my.yahoo.com. [31]
Portal Framework	Portal solutions often divide various portal-related components into layers, for example layers related to presentation, infrastructure, identity management. Portal products implement these components into an architecture specific to the product; the framework defines the general scope and relationships between the various components. There are often many similarities between the frameworks adopted across products (e.g. similar layers and components), although the specific implementations will vary.
WSRP (Web Services for Remote Portlets)	Web Services for Remote Portlets A standard that can be used to integrate remote portlets into a portal. WSRP, following a Web services path, is platform agnostic and can be used to present services through any WSRP-conformant portal [3] WSRP defines a Web Services interface for accessing and interacting with interactive presentation-oriented web services. Mark-up (e.g. HTML) is passed back to the consumer of the web service embedded in the response.
JSR168	A Java Community Process standard, Java Specification Request (JSR) 168, which describes a common method of rendering a 'portlet' (a portal component sometimes also referred to as a 'channel') within a Java-based portal framework. [3] Programmatically, the standard defines a contract (a set of APIs) between the portal and the portlets, JSR168 portlets are Java-only, and are local to the hosting container.
Portlet container	A layer within a portal framework that sits between a portal and its portlets providing a run-time environment for portlets and managing interaction between the portal and portlets.

11 Appendix B: Portal Products

Product Name	Company/ Organisation responsible	Open source/ Commercial	JSR168 Java	WSRP	Short Description and URL
Pluto/WSRP4J	Apache	OS - free	yes	yes	http://portals.apache.org/pluto/ Pluto is the reference implementation of JSR168, the Java portlet specification. Pluto serves as a portlet container that implements the Portlet API and offers developers a working example platform from which they can test their portlets. The project comes with a minimal portal for testing. The WSRP4J project provides the WSRP4J Producer, which allows implementing WSRP compliant services based on a free, open source software stack consisting of Tomcat, Axis and WSRP4J, which in turn includes Pluto. In addition, the WSRP4J project provides a generic proxy portlet written to the Portlet API, the WSRP4J Consumer. See sections 6.2 for further details.
EXo platform	EXoPlatform SARL	OS - Free GPL or commercial license	JSR168	No	http://www.exoplatform.org/portal/faces/public/exo The eXo platform software is a powerful Open Source corporate portal and content management system. The components include a portlet container which is a certified implementation of JSR168. Two versions are available, *express* which includes administration portlets, and technology wrappers for building Velocity, Struts, Cocoon or a Java Server faces portlets, and an IFrame portlet that allows the introduction of another web application in the portal, making it possible to use PHP, ASP or CGI applications in Java portlets. The *enterprise* version comes with its own application server and workflow management tools. The product comes with a content management system and a services container. An Eclipse plug-in is available for Java application developers, providing wizards and tools to assist portlet development.
GridSphere	GridLab project	OS and free (download	JSR168	No	http://www.gridsphere.org/gridsphere/gridsphere The GridSphere portal framework provides an open-source portlet based

	(funded by EU under IST)	requires email registration)			Web portal. GridSphere enables developers to quickly develop and package third-party portlet web applications that can be run and administered within the GridSphere portlet container. Here you will find the GridSphere portal framework available for download and documentation related to the installation and development of portlets using GridSphere.
UPortal	JA-SIG	OS - free	Supports JSR168 portlets from version 2.3 (using Apache's Pluto container)	From version 2.2	http://www.uportal.org/ uPortal is a free, sharable portal under development by institutions of higher-education. It is an open-standard effort using Java, XML, JSP and J2EE. It is a collaborative development project with the effort shared among several of the JA-SIG member institutions. It is a framework for producing a campus portal, not intended to be an out-of-the-box or "turn key" portal "solution". Presented as a set of Java classes and XML/XSL documents that you can use to produce a portal for use on your campus.
Jetspeed 2	Apache Jakarta	OS - free	JSR 168	Using WSRP4J	http://portals.apache.org/jetspeed-2/ Jetspeed is an Open Source implementation of an Enterprise Information Portal, using Java and XML. Jetspeed-2 is n Beta version and is conformant to the Java Portlet Standard. Jetspeed provides support for templating and content publication frameworks such as Cocoon, WebMacro and Velocity. See section 6.2 for moe details.
Liferay Enterprise Portal	Liferay	OS - Free	JSR168	Yes	http://www.liferay.com/products/index.jsp Liferay portal is designed to deploy portlets that adhere to the Portlet API (JSR 168). A number of portlets are bundled with the portal. The product is independent of application servers (can use Tomcat or Oracle or others) Any JSR 168 compliant portlets added should be available to consumers as WSRP.
Oracle AS Portal	Oracle	Some portal components	Java JSR168	WSRP	http://www.oracle.com/technology/products/ias/portal/product_overview.html

		are free e.g. Portal Development Kit but these work within the Oracle As portal A free Java portlet container is available.			http://www.oracle.com/technology/products/ias/portal/pdk.htmlOracle Application Server Portal provides a framework for integrating content from external sources. The external content is displayed to the user as windows on a portal page. Many functions within Oracle Application Server Portal itself are implemented as portlets. A number of different components are available, and the Portal Development Kit (PDK) comes with a Portlet Container for building and running interoperable Java portlets. The container provides a runtime environment for Java portlets coded to the standard Java Portlet Specification (JSR 168) APIs that enable the portlets to be utilized by any portal supporting the OASIS Web Services for Remote Portlets (WSRP) 1.0 standard. Portlets deployed to Oracle's Java Portlet Container are exposed automatically through WSRP An extension for JDeveloper provides a wizard for the step-by-step creation of portlets. The PDK enables developers to build portlets in any web accessible language including Java/J2EE, Web Services, PERL, ASP, PL/SQL, XML and much more.
Sun Java System portal Server 6	Sun	commercial	yes	yes	http://www.sun.com/software/products/portal_srvr/home_portal6.xml A Java portal that works with a number of application servers, provides additional development tools and utilities, provides single sign-on for aggregated applications to the portal, supports the creation and consumption of Web services-based portlets and incorporates the J2EE platform.
Vignette V7 Portal Services	Vignette	commercial	JSR 168	Not explicitly stated in the	http://www.vignette.com/contentmanagement/0,2097,1-1-1928-4149-1966-4151,00.html Vignette comes as an application portal (portal framework) and a builder for creation, assembly and customisation of applications. Pre-defined

				product web pages but Vignette supported the developm ent of the WSRP standard	portlets are available with the portal, including one for integrating .NET Web applications as portlets. The builder is intended to support portlet development in a 'code-free' development environment.
WebSphere Portal and Portal Toolkit	IBM	Free trial of the portal toolkit available for testing with the application server	Java	Not explicitly stated in the product web pages but IBM supported the developm ent of the WSRP standard	http://www-106.ibm.com/developerworks/websphere/zones/portal/bigpicture.html http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=products/portaltoolkit The IBM Portal Toolkit, Version 5.0.2.2/5.0.2.3 provides the capabilities to customize, create, test, debug, and deploy individual portlets and Web content. Before the creation of JSR 168, IBM had provided a proprietary API within WebSphere Portal. However, with the advent of JSR 168, it is recommended that portlet developers use the new standardized portlet API. The Portal Toolkit plugs into the IBM WebSphere Studio products. The foundation of the platform is IBM WebSphere Application Server. Every Application Server configuration is powered by a single Java™ engine, For development needs, IBM WebSphere Studio brings you a suite of tools in configurations that span development for the Web, the enterprise, and wireless devices. The WebSphere Studio development environment is based on the Eclipse Platform, an open universal platform for tools integration.
WebLogic Portal	BEA	Free	Java/JSR1	WSRP	An enterprise portal platform for production and management of custom-

8.1		Devlopment license (requires registration)	68 Supports the creation of JSR168 portlets but also has own portlets with extended features.		fit portals. Provides portlet wizards for the creation of different portlets (JSP/HTML, JSR168, Struts, WSRP) http://dev2dev.bea.com/products/wlportal81/index.jsp http://dev2dev.bea.com/products/wlportal81/articles/wsrp.jsp
Plumtree	Plumtree software	commercial	JSR168	WSRP	http://www.plumtree.com/developers/standards/default.asp The portal products are made available as part of an Enterprise Web Suite which also includes Content and Search servers. Portlet development is aided by wizards and graphical interfaces (rather than direct coding). Integration with J2EE and .NET
BowStreetPortlet Factory	Bowstreet	commercial	JSR168	Not stated	http://www.bowstreet.com/toolsandtechnology/portletfactory/jsr168.html Tools for the portlet development process, particularly for creating, customizing, maintaining, and deploying JSR-168 compliant portlets, masking the complexities of the underlying standards, including JSR-168 and J2EE.
Clickmarks PortletFactory	Clickmarks	commercial	JSR168	WSRP	http://www.clickmarks.com/index.html Provides portlet creation tools for Rapid Application Development, mainly for commercial portals such as Plumtree and Sun Java system. Claims support for JSR168 and WSRP.
Kapow	Kapowtech	commercial	Not stated	Not stated	http://www.kapowtech.com/solutions_portalprojects.htm Marketed as middleware to portal-enable legacy applications (Java), mostly for enterprise portal vendors

12 References

[1] Pearce, L. Institutional Portals: A Review of Outputs. Workpackage 3, Deliverable 1 of the PORTAL Project. July 2003.
www.fair-portal.hull.ac.uk/downloads/iportaloutputs.pdf
[2] The JISC Portals FAQ
http://www.jisc.ac.uk/index.cfm?name=ie_portalsfaq
[3] Awre, C., Dovey, M. J., Hunter, J., Kilbride, W. and Dolphin, I. Developing Portal Services and Evaluating How Users Want to Use Them: The CREE Project. *Ariadne*, Issue 41. October 2004.
http://www.ariadne.ac.uk/issue41/awre-cree/#36
[4] Thompson, R. and Schaeck, T. Enabling Interactive, Presentation-Oriented Content Services Through the WSRP Standard. XML Conference and Exposition 2003. Philadelphia, USA. December 2003.
http://www.idealliance.org/papers/dx_xml03/papers/04-06-05/04-06-05.html
[5] Kleane, M. Understanding the Java Portlet Specification. *Developer.com*. June 2004
http://www.developer.com/java/web/article.php/3366111
[6] Wilson, S., Blinco, L. and Rehak, D. Service Orientated Frameworks: Modelling the infrastructure for the next generation of e-Learning Systems. July 2004
http://www.jisc.ac.uk/uploaded_documents/AltilabServiceOrientedFrameworks.pdf
[7] Kropp. A, Leue, C and Thompson, R. Web Services For Remote Portlets Specification. OASIS. August 2003.
http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf
[8] A Day in the Life of a Software Developer: Portlet Development & Portals a weblog by Punit Pandey
http://blogs.ittoolbox.com/km/portals/
Also available at http://jroller.com/page/portlets and http://portlets.blogspot.com/
[9] The Yahoo! mailing list on portlets and related technology
http://groups.yahoo.com/group/portlets/
[10] Uncommented Bytes a weblog by Jeff Sheets
http://uncommentedbytes.blogspot.com/
[11] The Java.net Portlet Community site
http://community.java.net/portlet
[12] Blog by Ken Ramirez
http://weblogs.java.net/blog/ken_ramirez/
[13] Hepper, S. and Hesmer, S. Introducing the Portlet Specification, Part 1. *JavaWorld*. August 2003.
http://www.javaworld.com/javaworld/jw-08-2003/jw-0801-portlet.html
[14] Hepper, S. and Hesmer, S. Introducing the Portlet Specification, Part 2. *JavaWorld*. September 2003.
http://www.javaworld.com/javaworld/jw-09-2003/jw-0905-portlet2.html
[15] Kleane, M. Developing to the Java Portlet Specification. *Developer.com*. June 2004.
http://www.developer.com/java/web/article.php/10935_3372881_1
[16] BEA WebLogic Workshop Help (Online)
http://e-docs.bea.com/workshop/docs81/doc/en/core/index.html
[17] Shu, C. and Sum, M. Building JSR 168-Compliant Portlets with Sun Java Studio Enterprise. May 2004.
http://developers.sun.com/prodtech/portalserver/reference/techart/portlets.html

[18] Paeffagen, F. and Rick, B. Converting the WorldClock portlet from the IBM Portlet API to the JSR 168 portlet API. IBM developerWorks. December 2004. http://www-128.ibm.com/developerworks/websphere/library/techarticles/0412_paeffgen/0412_paeffgen.html
[19] Resources for Java server-side developers
http://www.java201.com/resources/browse/25-all.html
[20] The PORTAL project
http://www.fair-portal.hull.ac.uk/
[21] The SPP Project
http://www.portal.ac.uk/spp/
[22] JISC Portlas Programme Case Studies
http://www.jisc.ac.uk/index.cfm?name=project_portal_casestudies
[23] uPortal UK users' workshop, 19th November 2002
http://www.fair-portal.hull.ac.uk/19nov02.html
[24] Dovey, M. Rough Notes on implementing WSRP4J in uPortal
http://www.oucs.ox.ac.uk/portal/developers/environment.xml
[25] The JSR168 Specification
http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/
[26] Stone, R. JSR-168 and WSRP: Competitors or Partners TheServerSide.com April 2004
http://www.theserverside.com/reviews/thread.tss?thread_id=25598
[27] Linwood, J. and Minter, D. Building Portals With the Java Portlet API (Expert's Voice) Apress. August 2004
Sample Chapters available at
http://www.theserverside.com/articles/content/BuildingPortals/Sample+Booklet_Linwood-Minter.pdf
[28] The Pluto home page on the Apache web site
http://portals.apache.org/pluto/
[29] WSRP4J hom epage on the Apache web site
http://ws.apache.org/wsrp4j/
[30] Jetspeed 2
http://portals.apache.org/jetspeed-2/
[31] Morgan, E.L. An introduction to the Search/Retrieve URL Service (SRU). *Ariadne*. Issue 40. July 2004. http://www.ariadne.ac.uk/issue40/morgan/
[32] Davidson, C. and Coco, C. (Eds). Web Services for Remote Portlets 1.0 Frequently Asked Questions. WSRP-faq-1.0
http://www.oasis-open.org/committees/download.php/10953/