

Developing Grid Middleware with Enterprise Java Beans and Web Services Technologies

Xiaobo Yang
Software Developer
CCLRC e-Science Centre
Daresbury Laboratory
+44-1925-603791
x.yang@dl.ac.uk

Asif Akram
Software Developer
CCLRC e-Science Centre
Daresbury Laboratory
a.akram@dl.ac.uk

Rob Allan
Group Leader
CCLRC e-Science Centre
Daresbury Laboratory
+44-1925-603207
r.j.allan@dl.ac.uk

ABSTRACT

Web based portals are prevailing in Grid communities with the emerging Grid technology due to their ability to provide transparent access to Grid resources. Portal development has gained lots of momentum due to standardisation of portal technologies through the Web Services for Remote Portlets (WSRP) and Java Portlet Specification (JSR 168). In this paper, our recent experimental work done in the National Grid Service (NGS) Portal development will be discussed demonstrating the application of the J2EE technologies such as Enterprise Java Beans (EJB) and Web services in portal/portlet development. Mature J2EE standards for distributed and Grid computing are a natural choice due to their vast acceptance and support, whereas recently standardised portal specifications provide the missing presentation layer. Portals provide a convenient integration approach for diverse and geographically dispersed Grid resources, portal integration with process integration provides functionality enriched Grid middleware with a user friendly customisable and configurable presentation interface.

Categories and Subject Descriptors

D.1.3 [Programming Techniques]: Concurrent programming – *distributed programming*.

General Terms

Design, Security.

Keywords

portlet, portal, enterprise java bean, web service, grid.

1. INTRODUCTION

It has already been several years since the concept of “portal” was brought to the Grid community. While people are accustomed to think of portals as Web sites like www.yahoo.com

and www.ebay.com that integrate diverse information according to user requirements, portals have a special meaning to Grid users for their ability to provide transparent access to different dispersed Grid resources. Since the concept of Grid was first proposed in the mid-1990s, sophisticated middleware has been developed to hide the command line tools for executing Grid-related tasks. As Web browsers are ubiquitous nowadays, portals are a natural extension of Web based technologies like Java Servlet and JavaServer Pages, Active Server Pages and PERL etc. This makes portals automatically benefit from the abilities of Web browsers such as providing ease in use, pervasive, available world-wide, no installations hassles and no update conflicts.

The first-generation portals were based on middleware like GridPort (Grid Portal Toolkit) [1] and GPK (Grid Portal Development Kit) [2].

GridPort provides a collection of Perl modules to aid in the development of science portals on computational Grids. It is based on advanced Web, security and metacomputing technologies such as PKI (Public Key Infrastructure) [3] and Globus [4] to provide secure interactive services. On the server side, GridPort based portals exist as a set of Perl CGI scripts with different functional modules while on the client side users interact with simple HTML and JavaScript. GridPort has been adopted by many portal applications such as the Telescience Portal [5], the HotPage Grid Computing Portal [6] and the Cambridge CFD Grid Portal [7].

GPK is a package of Java beans that derive most of their functionality from the Globus Java Commodity Grid (CoG) Kits [8]. It provides five categories of Java beans: security, user profile, job submission, files transfer and information services. The NASA Launch Pad [9] was originally based entirely on GPK to provide Web based access to high-performance computer resources and NASA Information Power Grid (IPG) services.

Some other portals have their own libraries to support a project-specific user-interface. For example the BioSimGrid Web Portal [10] was built up on top of a set of Perl libraries called PortalLib for server side implementation.

As portals focus more on the presentation layers, the concept of “portlet” was introduced. Portals based on portlet technology provide the ability for user interface customisation, caching, persistence, and user authentications and authorisation (normally

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MGC 2005, November 28, 2005, Grenoble, France.
Copyright 2005 ACM 1-58113-000-0/00/0004...\$5.00.

role based access control). Developed on Java CoG and Jetspeed 1.0 [11] as building blocks, an experimental Grid portal [12] has demonstrated benefits of using the portal technology. The first release of the NGS portal [13] in 2004 was based on CHEF, a comprehensive collaborative framework originally developed at the University of Michigan (CHEF is now replaced by Sakai [14], a project for building and deploying a new Collaboration and Learning Environment (CLE) for higher education), CHEF also makes use of portlet-like technology for its tools. These tools or vendor-tied portlets can be classified as first-generation portlets, without any widely accepted standard, and therefore are portal container dependent.

Due to the lack of standards for first-generation portal/portlet development, the same portlets were built many times to support the APIs of different portal vendors. The second-generation portals are based on the two standards ratified in 2003 – WSRP [15] and JSR 168 [16]. Both standards were proposed to solve the interoperability issues in portal/portlet development. WSRP was officially ratified as an OASIS standard in September 2003. It is a Web services standard that allows portals (WSRP consumers) to easily embed portlets deployed in other portals (remote portlets published by WSRP publishers). Also WSRP has given new dimension to portal development and its effectiveness by providing a standard means to access portlets from non-Web and non-HTTP environments [17], and even to aggregate/integrate portlets from different portal containers into one application. JSP 168, published in October 2003 by the Java Community Process (JCP), was proposed to standardise portlets within one portal framework. In general, WSRP is a universal communication protocol between portals while JSR 168 is a set of Java APIs addressing the areas of aggregation, personalisation, presentation and security between portlets and portals. Therefore WSRP and JSR 168 are complementary to each other.

In this paper, we will first talk about the service-oriented architecture (SOA). Then we will move to portal/portlet development with J2EE technologies such as Enterprise Java Beans and Web services. Rather than developing Web services from scratch, stateless session beans are utilised to be exposed as Web services in our example. Finally, conclusions will be drawn based on our experiences followed by our plan of future work.

2. Service Oriented Architecture

Software architectures have attempted to deal with increasing levels of software complexity. Nevertheless the level of complexity continues to increase and traditional architectures seem to be reaching the limit of their ability to deal with the problem. SOA is being promoted in the industry as the next evolutionary step in software architecture to help IT organisations to meet their ever more complex set of challenges by enabling the assembly of applications through parts, components, or services. Although service-oriented architecture is not a new concept, it has been a hot topic since the late 1990s. The basic idea of SOA is to build up software applications by promoting loose coupling among distributed services so that reuse of services (components) is possible. Three of the most popular techniques used for building up SOA systems were Microsoft's Distributed Component Object Model (DCOM) (replaced by .NET Remoting in the .NET Framework), OMG's

Common Object Request Broker Architecture (CORBA) and Sun's Java Remote Method Invocation (Java RMI). Today, with their growth in popularity, Web services have become the mainstream technology for building up SOA systems.

SOA with a set of Web services specifications is best suited for development of systems where heterogeneity is fundamental to the environment, because they must accommodate an endless variety of hardware, operating systems, middleware and data stores. Middleware based on Web Services provides the best, realistic and practical approach for integration of heterogeneous resources which involves re-use rather than replacement of legacy code. This may be due to constrained budgets, effectiveness of legacy code for particular job, tight time schedule and dynamic nature of Grid computing particular in e-Science projects.

e-Science projects are diverse in nature and at present self-contained application systems are seen everywhere. These application systems may have been an excellent prototype design, but each of them is produced by and for a different line of business within the e-Science by a separately funded, isolated pilot projects. For example, the function of secure login based on MyProxy [18] credentials is re-implemented again and again in many different projects, even if they access the same account data in the same database; new projects add to the problem of all the redundant programming already in place, unless somehow the existing code can be reused. This redundancy can be attributed to the selection of different hardware, operating systems, programming languages and people. Web Services have removed one barrier by interconnection of applications in an object-model-neutral way, using a simple XML-based messaging scheme. The invoking application need have no idea where the business logic will be executed, what language it is written in, or what route the message may take along the way; a service is requested and answer is provided.

While there are many standards/technologies to realise Web services, in this paper we focus on Java J2EE 1.4 Web Services (Java Web Services can be either Apache Axis compliant or J2EE 1.4 compliant Web Services). J2EE can be treated as a robust suite of middleware services for server side application development. J2EE is a component-oriented architecture (COA) which can be extended to utilise the benefits of SOA. Distributed components such as Enterprise Java Beans are deployed and executed in J2EE application servers for local and remote access. When we talk about SOA, the key concept is that of service. A service can be defined as a self-contained software module, which can be a group of related (J2EE) components, that carries out a given business process. According to this definition, services in fact provide a higher level of abstraction from a functional point of view. With the prevalence of Web services, the J2EE 1.4 specification provides built-in support for Web services through JAX-RPC. JAX-RPC APIs are used for handling SOAP messages between Java and non-Java platforms so that plain old Java objects and stateless EJBs can be published as Web services. In general, SOA is not a replacement of COA, but a complement to COA. While COA is focusing on reusability at the component level, SOA enhances reusability at the service level, which is usually on top of components.

Although having similar to components to COA, SOA is also data-oriented. For example, session beans and entity beans are used for model business logic and persistence layers. Similarly, a Web service is normally used to execute tasks like retrieving Grid resource information. In multi-layer applications as shown in Fig 1, besides the business logic/service and persistence layer, there is normally a need for the presentation layer which interacts with end users. Many component-based Web development frameworks have grown to fit this demand, for example, the Apache Cocoon project [19], the Apache Struts framework [20], JavaServer Faces and the Jakarta Tapestry framework [21]. These frameworks normally provide a flexible control layer based on standard technologies like Java Servlet, JavaServer Pages and implement the classical Model-View-Control (MVC) design paradigm. Today's portals/portlets (second-generation) are designed with inborn support of the MVC design pattern, but as a whole they belong to the presentation layer in COA/SOA systems. Acting as a gateway between the users and services, portals provide the built-in functionalities to support personalisation, single sign-on (SSO), aggregation, customisation, etc. With data sources like EJBs and Web services, portals are ideal for presenting data to end users through Web browsers.

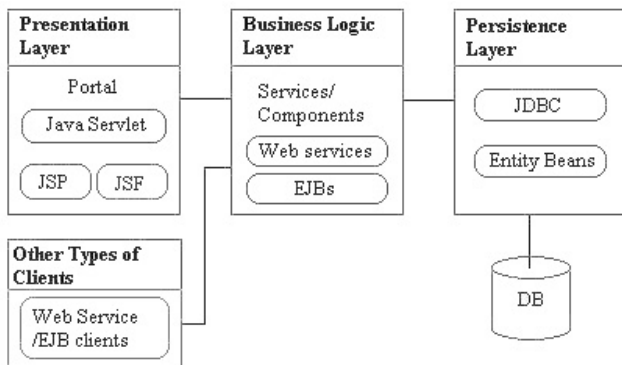


Figure 1. COA/SOA with portal acting as presentation layer

3. PORTLET DEVELOPMENT WITH J2EE

3.1 Integration Layer

Grid portals are becoming a predominant method of providing simplified access to diverse sets of resources in a Grid environment and portlet are being widely adopted in building Grid portals. Most of Grid functions require the user to access information that resides in multiple disparate systems, for example the user might need to transfer files using GridFTP [22]; however before GridFTP can be used he/she has to be authenticated. Switching between systems makes the overall workflow tedious and error-prone. There are different possible ways to integrate heterogeneous resources in the Grid environment - entity aggregation, process integration, data integration and portal integration. In our purposed solution we have combined process integration and portal integration to enhance the user experience.

3.1.1 Process Integration

Process integration can automate tasks that combine data and functions from multiple systems, thereby alleviating the need for users to access multiple systems directly. However process

integration requires good understanding of the business process so that the business process can be accurately represented in a process model. Process integration may not be the ideal solution as this gives less flexibility to the user and ties him to the business process and resources incorporated in the "Workflow".

3.1.2 Portal Integration

Portal integration is comparatively simple form of integration and less intrusive than more sophisticated types of integration. With portal integration, the business process that describes the sequence of tasks does not have to be represented in a precise process model, but instead provides different options to the user to select them at run time. Portal integration enables the end user to compensate for mismatches in semantics between the individual systems and to make a case-by-case decision. This approach is naturally less efficient than process integration, but it gives the user flexibility and it is typically much faster to implement. Portal integration can be used as an intermediate solution until business processes are better understood and can be incorporated into a process model and workflow.

3.1.3 Hybrid Solution

By combining process integration and portal integration we aim to bring the best of both worlds. Through process integration developers can combine different resources for process model in more efficient and robust way which leads to an application-oriented approach. Portal integration gives users access to different resources at a single location and provides single application interaction for disparate systems, which itself is a user-oriented approach. Most of Grid services are application-oriented, but at the same time consistent user interfaces are needed, thus requiring hybrid approach of combining process integration and portal integration.

3.2 NGS Portal with Enterprise Java Beans

A set of JSR 168 compliant portlets have been developed in the Grid Technology Group at the CCLRC e-Science Centre (Daresbury Laboratory) to provide UK National Grid Service (NGS) users access to its compute and data-nodes. The current release of the NGS portal includes proxy manager, job submission, job monitor, file transfer (through GridFTP [22] and SRB [23]) and LDAP/MDS [24] query portlets with built-in portlets from StringBeans [25], an open-source JSR 168 compliant portal framework.

As mentioned in our earlier work [26], we have successfully implemented EJBs in portal/portlet development. The clear separation of data sources from portlets is achieved using portals with J2EE 1.4 with EJBs handle the business tasks and persistence and portlets do the presentation part. As described in [26], our business logic is implemented in stateless session beans and Container-Managed Persistent (CMP) entity beans are utilised to talk to relational database tables.

As shown in Fig 2, we have three stateless session beans, `NgsUsersSessionFacade`, `NgsUserJobsSessionFacade` and `LdapQuery` with two entity beans, `NgsUsers` and `NgsUserJobs`. The two entity beans are used to map the two database tables, `ngs_users` for storing user credentials and `ngs_user_jobs` for storing job information users submitted. Three portlets - Proxy Manager, Job Submission and LDAP Browser have been

integration and portal integration and the boundary for middleware becomes blurred. For end users the portal is the middleware, thus the portal container constitutes middleware, but applications are independent of presentation layer and it is the Web service/session bean which is gluing underlying components together, e.g. MyProxy Server, database for persistence etc. Thus Web service and Web service engine are also part of the middleware. In such a hybrid solution we believe the presentation layer (portal/portlet) and business layer both are part of middleware; in most of the cases the portal container is integrated as part of the application server along with the Web container, but this is not necessary.

Things become more blurred with the introduction of WSRP [27]. The WSRP specification defines the following actors within WSRP-based SOA (see Fig. 3):

- 1) WSRP Producer: Producers are modeled as containers of portlets. The producers are Web services with a common set of operations such as: self description, mark up, registration, and portlet management. Producers can optionally manage the registration of consumers and require them to pre-register prior to interacting with portlets. A registration establishes a relationship between consumers and producers. The WSRP producer is a true Web service, complete with a WSDL and a set of endpoints. Every producer in WSRP is described using a standardised WSDL document which can be referenced from a UDDI registry.
- 2) WSRP Portlet: A WSRP portlet is a pluggable user interface component that lives inside of a WSRP producer and is accessed remotely through the interface defined by that producer. A WSRP portlet is not a Web service in its own right (it cannot be accessed directly, but instead must be accessed through its parent producer).
- 3) WSRP Consumer: This is a Web service client that invokes WSRP Web services offered by producers and provides an environment for users to interact with their portlets. Consumers are similar in nature to routers that work on behalf of the end user. The consumer will route requests from users to the appropriate Producer. The WSRP specification adds WSRP consumers as another component in the middleware layer. WSRP consumers interact with one or more WSRP producers thus acting as a glue agent in the portal environment.

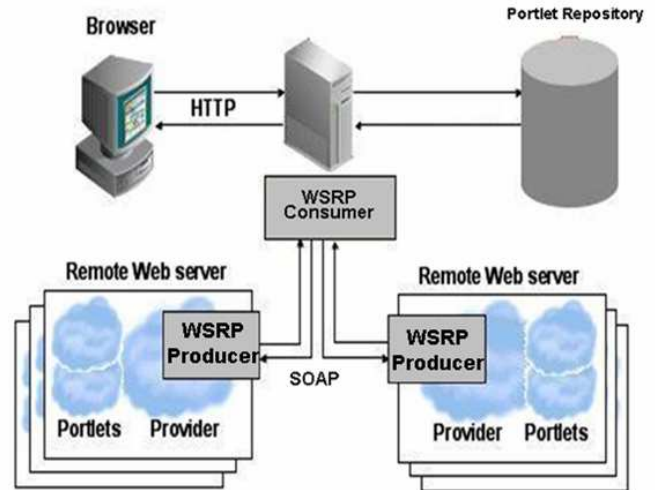


Figure 3. WSRP scenario.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we talked about applying Enterprise Java Beans and J2EE Web services in portal/portlet development. Adoption of EJBs for business logic and persistence has many benefits such as reusability, scalability and transactions which are provided by an EJB container. Furthermore, J2EE 1.4 supports Web services which make it easy to construct service-oriented architecture based on component-oriented architecture along with value added features provided by the application server.

Exposing session beans as Web Services is essential for integration in a heterogeneous environment. This can also be consumed by non-HTTP clients (Java/non-Java). Web Services can be registered in the local UDDI which will work as our portlet repository. Publishing Web service based portlets in UDDI will facilitate automated discovery, selection and consumption of portlets in different e-Science projects, further reducing the problem of redundant programming. The main purpose of this future work is to establish a collaboration between portal and Grid services developers sharing the registry.

Portals are convenient integration approaches for diverse and geographically dispersed Grid resources. Portal integration with process integration provides functionality-enriched Grid middleware with a user-friendly customisable and configurable presentation interface.

In the future, we would also like to extend our investigation on WSRP in an SOA since current support of WSRP in portal frameworks is still at an early stage as we have found in our initial research [27, 28].

6. ACKNOWLEDGMENTS

We thank the UK e-Science Core Programme for funding some of the work on the NGS Portal via the Grid Operations and Support Centre, JISC for funding work on the GROWL and Sakai Demonstrator projects via its VRE Programme and EPSRC for funding some work via a grant for Workflow Optimisation Services for e-Science.

7. REFERENCES

- [1] GridPort, <https://gridport.npaci.edu/>.
- [2] Novotny, J. The Grid portal development kit, *Concurrency Computat.: Pract. Exper.*, 2002, 14:1129-1144.
- [3] Public-Key Infrastructure (X.509), <http://www.ietf.org/html.charters/pkix-charter.html>.
- [4] Globus Alliance, <http://www.globusalliance.org/>.
- [5] Peltier, S.T., Lin, A.W., Lee, D., Mock, S., Lamont, S., Molina, T., Wong, M., Dai, L., Martone, M.E., and Ellisman, M.H. The Telescience portal for advanced tomography applications, *J. Parallel Distrib. Comput.*, 63 (2003), 539-550.
- [6] HotPage Grid Computing Portal, <https://hotpage.npaci.edu/>.
- [7] Yang, X., Hayes, M., Jenkins, K., and Cant, S. The Cambridge CFD Grid portal for large-scale distributed CFD applications, In *International Conference on Computational Science 2004 (ICCS2004)*, eds. Buak, M., et al., Springer-Verlag, LNCS 3036, 478-481, 2004.
- [8] Globus Java Commodity Grid Kits, <http://www.globus.org/cog/>.
- [9] NASA Launch Pad Portal, <http://www.ipg.nasa.gov/ipgusers/power/>.
- [10] Wu, B., Dovey, M., Hong Ng, M., Tai, K., Murdock, S., Fangohr, H., Johnston, S., Jeffreys, P., Cox, S., Essex, J.W., and Sansom, M.S.P. A Web/Grid portal implementation of BioSimGrid: a biomolecular simulation database, *J. Digital Information Management*, 2, 2 (2004), 74-78.
- [11] Jetspeed, <http://portals.apache.org/jetspeed-1/>.
- [12] Yang, X., Hayes, M., Jenkins, K., and Cant, S. The Cambridge CFD Grid for large-scale distributed CFD applications, *Future Generation Computer Systems*, 21, 1 (2005), 45-51.
- [13] Yang, X., Chohan, D., Wang, X.D., and Allan, R. A Web portal for the National Grid Service, *UK e-Science AHM2005*, Nottingham, UK, 2005, accepted.
- [14] The Sakai Project, <http://www.sakaiproject.org/>.
- [15] Web Services for Remote Portlets (WSRP), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp.
- [16] JSR 168: Portlet Specification, <http://www.jcp.org/en/jsr/detail?id=168>.
- [17] Akram, A., Chohan, D., Wang, X.D., Meredith, D., and Allan, R. CCLRC portal infrastructure to support research facilities, *GGF Workshop on Science Gateways, GGF14*, Chicago, IL, USA, June 2005.
- [18] Novotny, J., Tuecke, S., and Welch, V. An online credential repository for the Grid: MyProxy, In *Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, 2001, 104-111.
- [19] The Apache CoCoon Project, <http://cocoon.apache.org/>.
- [20] The Apache Struts Web Application Framework, <http://struts.apache.org/>.
- [21] Jakarta Tapestry, <http://jakarta.apache.org/tapestry/>.
- [22] Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., and Foster, I. The Globus striped GridFTP framework and server, *Super Computing 2005 (SC05)*, Seattle, WA, USA, November 2005, to be published.
- [23] Storage Resource Broker, <http://www.sdsc.edu/srb/>.
- [24] Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., and Tuecke, S. A directory service for configuring high-performance distributed computations, In *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*, Portland, OR, USA, 1997, 365-375.
- [25] StringBeans Portal, <http://www.nabh.com/projects/sbportal>.
- [26] Yang, X., Akram, A., and Allan, R. Developing portal/portlets using enterprise java beans for Grid users, Submitted to *International Conference on e-Science and Grid Technologies*, Melbourne, Australia, 2005.
- [27] Akram, A., Allan, R., and Crouchley, R. WSRP reincarnation of service oriented architecture, *UK e-Science AHM2005*, Nottingham, UK, September 2005, accepted.
- [28] Akram, A., Chohan, D., Wang, X.D., Yang, X., and Allan, R. A service oriented architecture for portals using portlets, *UK e-Science AHM2005*, September 2005, accepted.