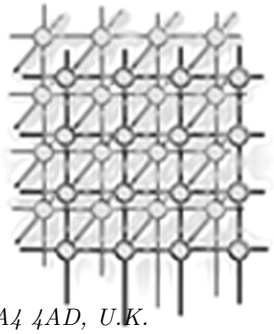
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2005; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Developing portal/portlets using Enterprise JavaBeans for Grid users



X. Yang*,†, A. Akram and R.J. Allan

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

SUMMARY

A web based portal acting as a presentation layer to the Grid has a means that we can provide transparent access to Grid resources. Our previous experience in developing Grid portals has shown that there is a need to draw a clear high-level picture of the architecture. This architecture could be defined using Enterprise JavaBeans (EJB), which enables a definate separation between the presentation, business logic and data layers. Additional benefits come with J2EE 1.4 which makes it possible to build up advanced service-based Grid portals by exposing stateless session beans as web services. In this paper, lessons learnt from our prototype portal/portlet development using EJBs are reported.

KEY WORDS: portal; portlet; enterprise javabeans, web services

1. INTRODUCTION

Web based Grid portals such as the Telescience Portal [1], the HotPage Grid Computing Portal [2] and the BioSimGrid Web Portal [3] are prevailing in Grid communities for their support of transparent access to Grid resources. Portals are essentially web pages that provide a centralised point for accessing resources which can vary from a simple application such as e-mail or a web search engine to a complex Grid application. Today portlets are widely adopted in constructing portals. These make extensive use of technologies including Java Servlet and JavaServer Pages (JSP) for their ability to create dynamic web content. The MVC (Model-View-Controller) programming pattern is widely utilised in portlet development. Both Struts [4] and JavaServer Faces (JSF) are also now supported by different portal

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

Received 22 December 2005
Revised ?? ?? 2005

frameworks and bring more sophisticated patterns. For example, the eXo Platform [5] uses bridges to support both Struts and JSF.

To solve the interoperability issues in portal/ portlet development, two standards – WSRP (Web Services for Remote Portlets) [6] and JSR 168 (Java Specification Request 168 Portlet Specification) [7] were proposed in 2003. As mentioned in [8], whilst WSRP is a universal protocol for communication between different portals, JSR 168 is a set of Java APIs that standardise communications between Java portlets and a portlet container. With these two specifications, portlet developers can concentrate on developing standard JSR 168 portlets while portal frameworks would provide the WSRP support to publish their portlets (as producers) and consume remote portlets (as consumers), i.e. portlets inside remote portlet containers. The JSR 168 specification is adopted by most of the portal frameworks today but we have found the WSRP support to be still immature [9]. While some of the portal frameworks provide WSRP consumers to consume remote portlets, almost none of their WSRP producers can be consumed by other portal frameworks without any issue. In addition to being published as remote portlets within a normal portal framework which has WSRP producer support, portlets can be published through 3rd-party WSRP producers like WSRP4J [10] which in turn makes use of Pluto [11] as its portlet container. As reported in [9], neither producers nor consumers are however fully functional. Therefore in this paper, only JSR 168 portlets are considered.

No matter how complex portals become, they are normally treated as a presentation layer to business logic which manages special features like customisation and single sign-on (SSO). With development experience in several portal projects such as the NGS Portal [12] and the Cambridge CFD Grid Portal [13], we believe that high-standard software development should be based on proven best practices, thus keeping the business logic and presentation layer separate. Simple portals can combine presentation and business logic for convenience, but it is hard to re-use the code. Our experience shows that for more sophisticated portals it is neither convenient during development nor later for maintenance. Business logic can be implemented either as some simple Java classes or complex J2EE components such as Enterprise JavaBeans [14]. Theoretically, selection of technology is governed by the requirements and complexity of business logic. Today Grid portals are becoming more and more complex in their functionalities and need persistence, security, transactional control, load balancing and much more. We therefore decided to investigate implementing business logic and data layers as EJBs.

J2EE developers often consider the use of EJBs to be a heavyweight technique owing to their complexity and in-efficiency. As a widely adopted industrial standard however, J2EE/EJB is quite mature and therefore we have explored how the J2EE architecture can benefit us in portal/portlet development. The web service support in the EJB 2.1 sepcification extends the flexibility of EJBs which are now no longer limited to RMI-IIOP. The new EJB 3.0 specification (JSR 220, now under public review) tries to solve the complexity of current EJB techniques by introducing the programming annotation facility of Java 5 and a new object-relational (O/R) mapping model based on Hibernate [15]. These new features of EJB plus the existing values of mature J2EE application servers such as scalability, transaction and security support make it a promising technology for our purpose.

In this paper, we will first describe the J2EE based component-oriented architecture (COA) with focus on using Enterprise JavaBeans in our prototype portal/portlet development work. Then we describe how to convert COA to a service-oriented architecture (SOA). This is followed by a description of the initial work on developing Grid portlets based on EJBs. Some related work on advanced serviced-oriented Grid portals is also presented followed by conclusion remarks.

2. COMPONENT-ORIENTED ARCHITECTURE AND SERVICE-ORIENTED ARCHITECTURE

2.1. COA

The J2EE architecture is an effective component-oriented architecture. The EJB technology, one of the key technologies in J2EE, provides the server-side component architecture for J2EE. EJBs model both business and data layers by defining session beans and entity beans. Without worrying about how to present information, session beans focus on how to manage data for users. This could involve complex algorithms, business procedures, etc. The use entity beans to manage objects in persistent storage and talk to them to either fetch data from or write data to rational databases. Especially with container-managed persistence, the heavy duty of database operations is now the task of the application server rather than programmers. Besides session and entity beans, a third-type of EJB, message-driven bean offers a robust and scalable enterprise messaging solution to leverage existing enterprise applications.

An extra presentation layer should normally be added to COA to provide user interfaces (UI), see Fig. 1. The nature of EJBs makes it possible to have multiple UI clients such as a desktop program or a web application. Today, web interfaces are widely adopted for e-business and e-science. J2EE provides Java Servlet, JavaServer Pages and JavaServer Faces to generate dynamic web pages for rendering information in web browsers. As mentioned previously, by extending the servlet specification, web pages managed by portals can now be composed of a set of web components called portlets.

2.2. SOA

Web services provide a useful way for building up a message-based Service-Oriented Architecture. The core idea of SOA is loose coupling among interacting software agents. A service sits on the service provider side to execute a set of data-centric tasks, the definition of which and the API is published. Because of this publishing mechanism a web service-based SOA is suitable for constructing large-scale applications that integrate heterogeneous distributed comonents. Although SOA is now commonly realised through web services, it can also be done using other techniques such as CORBA (Common Object Request Broker Architecture).

COA and SOA are in fact complementary architectures. A COA based system can be converted to an SOA based system in a straightforward way by exposing the EJBs as web services. The EJB 2.1 specification brings support for this through exposing stateless session beans as web services. Thus when we talk about COA in this paper, it can also imply SOA.

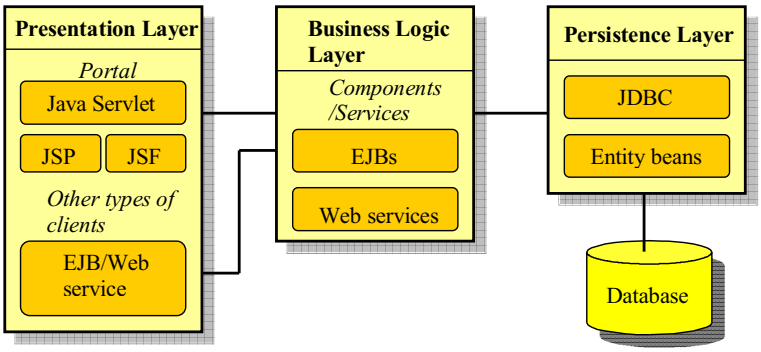


Figure 1. COA/SOA with portal acting as the presentation layer.

By adopting the loosely coupled COA, components can be platform-independent and can be easily shared, e.g. between the Java and Microsoft .NET communities. By re-using EJBs or remote services, portal/portlet developers can focus on how to present data rather than writing and placing complex business logic in the portlets. This greatly reduces the burden on portal developers in deploying large complex systems. The drawback here is that besides a servlet/portlet container, an extra EJB container is needed which makes the overall system more complex. EJB development and testing is also much more difficult compared with POJOs (Plain Old Java Objects). The benefit however is that once deployed the services can be re-used in many different portals to support a divers range of projects and end-user communities.

3. PROTOTYPE PORTAL EXAMPLE

3.1. NGS Portal

The example we are going to discuss is based on our early work on portal development for the UK National Grid Service (NGS) [12]. NGS is the UK's core production computational and data Grid which runs Globut Toolkit 2 (GT2) and other services including SRB [16]. StringBeans [17], an open-source JSR 168 compliant portal framework, has been adopted as the choice for the NGS Portal. It has been customised to accept authentication through a MyProxy [18] server which is maintained separately from the portal server. As StringBeans supports the Java Authentication and Authorisation Service (JAAS) pattern, a MyProxy login module has been written to realise our customised authentication through MyProxy. With this module, end users can now use Grid tools after successfully logging into the portal, since their Grid proxy credentials are retrieved automatically during the login procedure. Some generic

JSR 168 portlets were developed to support the generic functionality provided by Globus with Grid proxy management, job submission, job monitoring, file transfer (using GridFTP [19]) and LDAP/MDS (Metacomputing Directory Service) [20]. SRB is also supported. Currently, data providers are included in the portlets with database access based on Hibernate [15].

3.2. Example portal based on EJBs

In this section we will not cover all the portlets mentioned above but focus on MDS/LDAP query, proxy manager and job submission. The basic idea is to keep our Grid portlets as simple as possible, providing easily adaptable presentation layers to the Grid applications. The business logic has been removed from the portlets and is implemented in the session beans instead. In the demonstration portal application, users log in using their portal usernames and passwords, that is, the login procedure is managed by the portal. We did not use the MyProxy login module in this test, so there is no modification of the portal framework itself. Authenticated users are given access to different portlets by the portal according to their roles as specified by the portal administrator.

A set of session beans and entity beans have been written for portlets to consume. These include two container-managed persistent (CMP) entity beans each of them maps to a database table which maintains a user profile and records jobs submitted by the users. As we can see from Fig. 2, NgsUsers and NgsUserJobs are mapped to two tables, *ngs_users* and *ngs_user_jobs* in the left corner. These two entity beans are wrapped by two session beans for external access. An additional session bean, the LdapQuery bean, has been developed for retrieving information such as CPU type and memory size from the NGS MDS server. Similarly to portlets in the NGS Portal, these EJBs are based on Globus Java CoG 1.2.

Based on the EJBs mentioned above, we have developed three portlets for demonstration purposes. A LDAP Browser portlet for MDS query, a Proxy Manager portlet for retrieving/renewing Grid proxy credentials from the NGS MyProxy server and a Job Submission portlet for submitting jobs to a remote Globus GRAM Gatekeeper. These three portlets were developed from the portlets used in the NGS Portal with only minor modifications in replacing the built-in business logic and persistence parts with EJBs. The EJBs and portlets together with the StringBeans portlet container have all been successfully deployed in the JBoss Application Server [21]. Fig. 3 gives a screenshot of these three portlets running inside StringBeans.

The LdapQuery session bean has also been successfully exposed as a web service with the following steps applied:

(i) define an endpoint interface with all operations to be published in the web service. The endpoint also needs to be defined in the ejb-jar.xml configuration file;
(ii) create deployment descriptors including webservices.xml (which defines the set of web services that are to be deployed in a web services for the J2EE enabled container) and a JAX-RPC mapping file (which defines classes used for serialisation/de-serialisation of custom data types) with vendor-dependent configuration files;
(iii) re-compile and package everything including EJBs and deploy them.

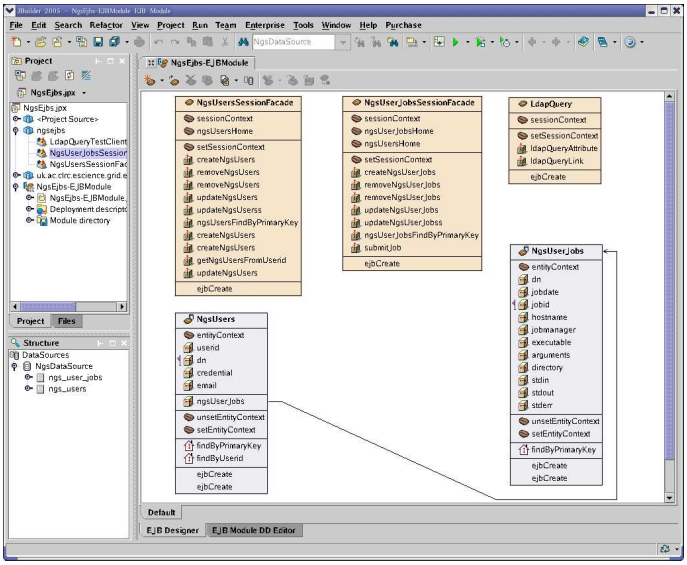


Figure 2. EJBs developed for our example portal.

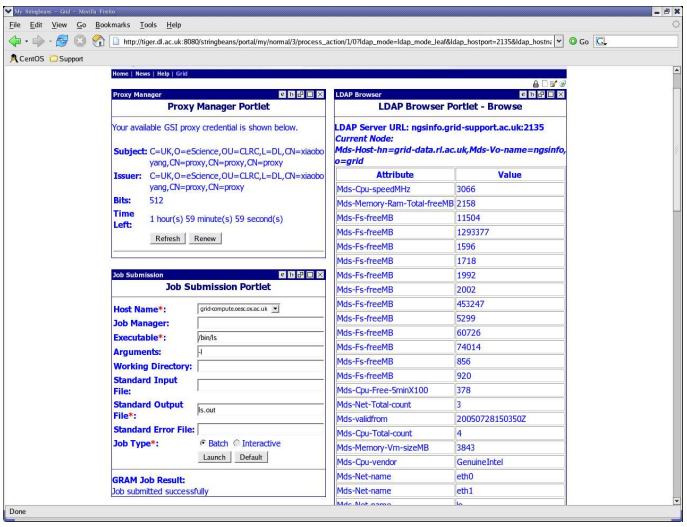


Figure 3. Screenshot of portlets running in StringBeans.

The core LDAP query functions are now available for non-HTTP clients to access which makes the whole system much more flexible to suite different scenarios. This demonstrates the flexibility of the J2EE architecture since from version 1.4 as it begins to support web service and the COA can simply become and SOA.

3.3. Hibernate vs. entity bean

From the programmers' point of view, it is always easier to understand and more efficient to handle objects rather than direct relational database queries. As mentioned in Section 3.1, Hibernate is utilised in the current release of the NGS Portal as the data persistence service. Hibernate, one of the O/R mapping technologies, is widely adopted for developing persistent classes in Java. It can either store Java objects in a relational database or provide an object-oriented view of existing relational data. Similar to EJB which has EJB Query Language (EJB-QL), Hibernate has its own portable SQL extension (HQL). Both techniques are attempts to simplify the data layer management for software developers. Unlike entity beans, Hibernate does not need a container but only some libraries which have to be included in the application. It is often argued in the Java community that CMP entity beans have disadvantages compared to O/R mapping technologies such as Hibernate:

(i) persistence technology within a container normally involves other container-based services which makes it complex;
(ii) entity beans are not portable between application servers;
(iii) without an IDE (Integrated Development Environment), CMP entity bean development can involve a lot of work.

Since the proposed EJB 3.0 specification (JSR220, now under public review) could make it a lot easier to developing EJBs, Hibernate is begining to support the EJB 3.0 persistence standardisation effort. In JBoss, it is now used as a container-managed persistence service.

For the persistence service, there is no longer much difference between entity beans and Hibernate, especially in EJB 3.0. Persistence is also only part of the EJB specification. The benefit of adopting EJBs lies mainly in other types of EJBs (session bean and message-driven bean) and additional functions provided by the application server. In fact, it is possible to develop J2EE applications without even using EJBs by adopting frameworks like Spring [22], but this is outwith the scope of this paper.

4. RELATED WORK

Many Grid portals are nowadays *service* based. Besides providing traditional functions, advanced service based Grid portals normally provide their functionalities as web services which can be consumed outside of the traditional portals/portlets in an SOA.

GridPort 3 [23] adopts a clear 3-tier architecture which includes a J2EE web server tier, a J2EE application server tier and a database and service tier. This is very similar to the J2EE 3-layer architecture illustrated in Fig. 1. In GridPort 4 the architecture has been re-designed with three layers called portal layer, service layer and resource layer. Compared to

the COA-based one, the latter places emphasis the importance of *serivce* and *resource*. For example, Grid related functions such as GridFTP and GRAM can be treated as generic services while database and HPC are included in the resource layer. Based on SOA, GridPort 4 thus provides users with a simple, easy-to-install software package for creating portals to access Grid infrastructure via a web interface.

NEESit [24] links earthquake researchers together across the USA with computing resources and engineering facilities. It provides a virtual research laboratory for data management, collaboration, telepresence and simulation. In order to link researchers effectively, web-based portals such as NEEScentral and NEESport [25] have been developed. By aggregating and coordinating lower-level services provided by the Globus Toolkit, NEESport provides high-level middle-tier services for seamless access to remote resources. In NEESport again the concepts *service* and *resource* are explicitly emphasised.

Another U.S. based example is the GEON (Geoscience Network) cyber-infrastructure project [26] which aims at creating an infrastructure to enable virtual laboratory for geoscience researchers. A service-oriented architecture is also adopted with a portal developed for end-users to access a set of portlets.

5. CONCLUSIONS

Current portals developed for Grid communities normally bind presentation, business logic and data sources within portals/portlets. In this paper, we discuss applying the J2EE pattern-based techniques especially the Enterprise JavaBeans to model business logic and persistence layers. EJBs are candidate data sources for portals/ portlets. The benefit of adopting mature J2EE techniques in portlet development brings a lot of added value such as reusability, scalability, security and transaction support which needs further investigation. With data providers extracted from portlets as EJBs, the structure of our portal system becomes very clear with a focus on presentation only. By using container-managed persistent entity beans, the J2EE application server handles data with automatic transaction support. By exposing stateless session beans as web services, the components like EJBs can be easily converted to web services enabling the building up of a suite of services for re-use in advanced service-oriented Grid portals serving divers user communities.

Whist a basic portal based on EJBs has been demonstrated successfully, this is only the starting point of adopting J2EE technology. As there have been arguments against using J2EE and EJB on grounds of complexity, this study has enabled us to identify the benefits and difficulties in the context of our own work. We are now beginning to plan how to use EJBs to build up much more complex systems than the prototype portal example. The final system envisaged will permit ongoing development and maintenance of business logic as a single set of code which can be used in several portal interfaces and also exposed as web services to be consumed in desktop applications requiring access to Grid resources such as the GROWL Virtual Research Environment (VRE) [27] project.

On the other hand, portals can be built up on top of existing legacy systems. If such a system happens to be constructs using EJBs, as shown in this paper, these systems can then be easily integrated in a portal.

ACKNOWLEDGEMENTS

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

REFERENCES

1. Peltier, S.T., Lin, A.W., Lee, D., Mock, S., Lamont, S., Ellisman, M.H. The Telescience Portal for Advanced Tomography Applications. *Journal of Parallel and Distributed Computing*, 2003; **63**(5):539-550.
2. HotPage Grid Computing Portal. https://hotpage.npaci.edu/ [1 December 2005].
3. Wu, B., Dovey, M.,Ng, M.H., Tai, K., Murdock, S., Fangohr, H., Hohnston, S., Jeffreys, P., Cox, S., Essex, J.W., Sansom, S.P. A Web/Grid Portal Implementation of BioSimGrid: a Biomolecular Simulation Database. *Journal of Digital Information Management*, 2004; **2**(2):74-78.
4. Struts. http://struts.apache.org/ [1 December 2005].
5. eXo platform. http://www.exoplatform.com/ [1 December 2005].
6. WSRP Specification 1.0 by OASIS. http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf [1 December 2005].
7. JSR 168: Portal specification. http://www.jcp.org/en/jsr/detail?id=168 [1 December 2005].
8. Vickers, S. Portal Standards the Answer to Portal Interoperability? *Java Developer's Journal*, 2005; **10**(1):18-20. 18-20.
9. Yang, X., Wang, X.D., Allan, R.J. WSRP Support Investigation of Selected Open-Source Portal Frameworks. GCE05, submitted to *Concurrency and Computation: Practice and Experience*.
10. WSRP4J. http://ws.apache.org/wsrp4j/ [1 December 2005].
11. Pluto. http://portals.apache.org/pluto/ [1 December 2005].
12. Yang, X., Chohan D., Wang X.D., Allan R.J. A Web Portal for the National Grid Service. In *UK e-Science AHM 2005*, Nottingham, UK, 2005; available on CDROM.
13. Yang, X., Hayes, M., Jenkins, X.D., Cant, S. The Cambridge CFD Grid Portal for Large-Scale Distributed CFD Applications. In *International Conference on Computational Science 2004 (ICCS2004)*, eds. M. Buak etc al., Springer-Verlag, LNCS 3036, pp. 478-481, 2004.
14. Enterprise JavaBeans Technology. http://java.sun.com/products/ejb/ [1 December 2005].
15. Hibernate. http://www.hibernate.org/ [1 December 2005].
16. Storage Resource Broker. http://www.sdsc.edu/srb/ [1 December 2005].
17. StringBeans. http://www.nabh.com/projects/sbportal/ [1 December 2005].
18. Novotny, J., Tuecke, S., Welch, V. An Online Credential Repository for the Grid: MyProxy In *Procceddings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, 2001; pp. 104-111.
19. Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., Foster, I. The Globus striped GridFTP Framework and Server. In *Proceedings of Super Computing 2005 (SC05)*, to be published.
20. Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., Tuecke, S. A Directory Service for Configuring High-Performance Distributed Computations. In *Proc. 6th IEEE Symposium on High-Performance Distributed Computing*, 1997, pp. 365-375.
21. JBoss Application Server. http://www.jboss.com/products/jbossas [1 December 2005].
22. Spring Framework. http://www.springframework.org/ [1 December 2005].
23. The GridPort Toolkit. http://gridport.net/ [1 December 2005].
24. NEESit. http://it.nees.org/ [1 December 2005].
25. NEESport: Simulation Portal for NEESgrid. http://www.erc.msstate.edu/ haupt/NEES_WEB/NEESport.html [1 December 2005].
26. Cyberinfrastructure for the Geosciences. http://www.geongrid.org/ [1 December 2005].
27. GROWL: Grid Resources on Workstation Library. http://www.growl.org.uk [1 December 2005].