Integrative Biology VRE

Work Package 7: Portal Technology
Evaluation Report

Geoff Williams

Intended Audience
Current and future portal technology development and user community.

Issued: 31 Mar 2007
1 Introduction

This document refers to the experience gained in the use of the latest stable release version series of uPortal i.e. 2.5.x, and in the use of JSR-168 (Portlet 1.0 specification).

It is a subjective, rather than a definitive or authoritative perspective.

Portals are generally considered to be web applications that aggregates content and provides the users with a personalized experience through the ability to express their preferences. The individual components of the portal content are the portlets which are grouped in a tabbed environment, and usually many portlets will appear on a single tab.

2 uPortal

2.1 Introduction

The following is taken from the JA-SIG (Java Architectures Special Interest Group) website, the group behind uPortal development, and the quote explains uPortal's raison d'être:

"uPortal is a free, sharable portal under development by institutions of higher-education. This group sees an institutional portal as an abridged and customized version of the institutional Web presence... a "pocket-sized" version of the campus Web."

uPortal has been under development for several years and is currently undergoing version 3 RC 1 (Release Candidate) testing of the software. The latest stable release version, at the time of writing, is version 2.5.3. In the early months of the IBVRE project a "version 3" of uPortal was available for download but after some weeks investigating the operation and functionality it was considered inoperable owing to lack of working features and instability. The decision to use version 2.5.1 for the IBVRE was made.

There are two general releases available for potential installation: The first is a "uPortal-only" source code distribution for compilation and installation in a production environment and for this the administrator will need to provide a servlet engine (e.g. Jakarta Tomcat) and relational database (e.g. Postgres). The second is a preconfigured, compiled "quickstart" distribution for an easy installation of all the software to get the relevant version of uPortal up and running with minimal configuration.

Being a Java application, uPortal will run on any platform which has a Java 2 implementation available for it.

Interesting to note is that uPortal's earlier implementations of what are generally now considered to be portlets are referred to as "channels" although this is being phased out in current/future development.

2.2 Configuration and Building

For the uPortal-only distribution the configuration and installation relies on the use of the Apache Ant build system. After downloading the uPortal source code in the distribution the following Ant tasks are available:

addstylesheet, all, clean, compile, compiletests, db, dbtest, dbunload, delstylesheet, deluser, deploy, deployPortletApp, dist, help, i18n-db, initportal, javadoc, l10n-db, md5passwd, modstylesheet, prepare, pubchan, pubfragments, regchantype,
runtests, RunXSLT, and swingtests.

Of these the important one is initportal which will run all the targets necessary to install uPortal and set up the database and initialise the data (deploy, db, i18n-db, pubchan, pubfragments).

For the purposes of the IBVRE it was decided to modify the uPortal build process so that we were able to create an IBVRE-specific portal environment as in many regards a virtual research environment is not reflective of an higher-education establishment. This would involve modifying the initialisation data of many of the uPortal entities such as the users, groups, portlets and access controls, so that irrelevant options, data and values would not appear in the VRE itself. This process, whilst providing the benefit of a much better understanding and insight of the internals of uPortal, was unfortunately very time-consuming. There appears to have been no prior need for this specialisation within the uPortal user community, and certainly there was neither documentation nor advice forthcoming from other users or developers as to how to go about implementing such a strategy. In the end the process involved extensive modifications of the configuration xml data files used to prepopulate the database as there wasn't even an option to create an empty yet operational database. These modifications need to be considered when version-upgrading uPortal as new versions of uPortal add functionality or resources which use values defined in the original default database configuration - which in the case of IBVRE may have been either removed or reassigned.

We also took the time to develop a user interface stylesheet based on one of the existing stylesheets shipped with the uPortal application.

2.3 Functionality

uPortal appears to have much of the functionality required of its intended audience such as single sign-on (SSO) and the ability to integrate alternative third-party authentication systems (e.g. Lightweight Directory Access Protocol - LDAP) as the "framework" nature of uPortal hides the implementation details for developers behind interfaces.

uPortal also provides:

a) User preferences and content control;

b) (Partly) Internationalised display;

c) Group and User management;

d) Permissions management;

e) A variety of existing "channels" (portlets) e.g. RSS feeds, News feeds, Inline frames for static HTML, and XSLT transformer implementations.

f) Limited Web Services for Remote Portlets (WSRP) functionality whereby one of the uPortal channels is able to act as a consumer of a WSRP producer channel.

g) Portlet caching to prevent unnecessary refresh of portlet content.

2.4 Operation

uPortal defines three levels of administration:

1: "administrative" Principally related to uPortal installation, visibility and uptime work e.g. web server and database work.
2 : "implementation" A more uPortal-specific administrative role, customizing uPortal for the site.

3 : "content development" Developing, incorporating and delivering content for the portal to the users.

2.5 +ves

a) Stable.

Although the IBVRE is only enduring very light use to date there haven't been any occurrences of uPortal crashing completely, although individual portlets have occasionally failed and required a uPortal restart. In the case of the IBVRE the portal wasn't directly receiving HTTP requests, the requests were instead handled by an Apache web server which passed requests through to Tomcat via a Jk connector.

b) Easy to operate.

Once the content administrator is familiar with uPortal then operation is generally straightforward although there are some areas where this isn't intuitive and the operational manuals are required. For the purposes of the IBVRE there wasn't the need to spend lots of time investigating all the configurational possibilities.

It is relatively straightforward as a uPortal administrator to deploy JSR-168 portlets into uPortal using the ant deployPortletApp build instruction. Once physically deployed into the servlet container the administrator then needs to go through the channel management process to publish the new content. Although portlets can be placed manually into the appropriate servlet container directory doing this will bypass the uPortalisation of the portlet application - principally the modification of the web.xml file.

c) Fragment pushing.

One of the options utilised in the IBVRE was the creation of "fragments" in the fragment manager. Fragments refer to the grouping of portlets and an ability to "push" or "pull" these fragments into the view of particular users or groups.

2.6 -ves

a) Documentation.

Documentation appears to be an area which uPortal developers/implementors have not appeared to dedicate much time to for whatever reason. Although there is a frequently used mail list to offer support and advice on issues ranging from basic operation to technical performance optimisation, as well as a JA-SIG collaborative wiki, there isn't always a straightforward route to relevant (i.e. up-to-date and detailed) online documentation which can be quite frustrating.

b) Access Control.

uPortal currently appears to be able to determine grouping of users via a model of demarcation of authority within a single organisational hierarchy e.g. students, lecturers, professors, etc. This however contrasts with the use in the IBVRE whereby groups may be derived from many people within different organisations, even restrictive within the same physical department, and at different authoritative levels. Fine-grained access control could be assigned on an individual-by-individual level but this would be inflexible and time-consuming.
c) Legacy Settings and Functionality Duplication.

Unsurprisingly in any application development new functionality arrives which supercedes existing functionality and in order to accomodate existing users in the upgrade cycle there needs to be backwards compatibility.

This appears in uPortal to have been particularly evident in the client-side display functionality, of which there appeared to be three different models (ALM, DLM and SLM) catered for simultaneously. It was a little disconcerting to think that three separate and incompatible options were present and it was difficult to determine from documentation which was the best solution to choose.

There also appeared to be a dependency on uPortal "channels" which were the original incarnation of what are now referred to as "portlets".

d) Redundant and residue data.

In some situations when all known references to uPortal entities, e.g. portlets, had been removed via the client/user interface their seemed to be residue data left in the database. Not only this, on system installation there seemed to be some data in the database which seemed to be redundant or irrelevant although how much of this could have been due to the specifics of the IBVRE install modifications wasn't determined. Although these did not appear to impact on performance or function the presence of this data indicated the possibility of lack of suitable housekeeping and/or development attention by uPortal.

e) UTF-8 character set use in the database.

Unfortunately uPortal's database definitions allow long character strings to be used as table indices which, in MySQL with a 1024 character limit, would not be allowed if the default character set of the database was set to UTF-8. This meant that for uPortal an alternative character set, latin-1, was used. Although this was not a real problem for the project it did highlight some uPortal design aspects, i.e. surrogate keys usage.

f) Spaces in names.

One problem encountered involved the creation of a user on the system which resulted in problems because the user name hadn't had preceding spaces trimmed prior to entry.

2.7 Future

In version 3 of uPortal, JSR-168 portlets will be the native content model rather than the present channel implementation.

Additional improvements will be the focus on providing better configuration flexibility and source code design. To achieve this the uPortal developers will use the Spring framework, Hibernate and other third party software which allow application developers to focus on creating maintainable and easily testable business logic.

3 Portlets

3.1 Introduction

Portlets represent the content of the portal and are recommended to conform to the JSR 168 standard in their development and operation. By conforming to this standard it is anticipated that such portlets will be able to be deployed in any portal application with perhaps only the
need for additional portal-specific configuration files or minor portal-specific configuration changes in the portlet configuration files.

3.2 Conformity to Standard

Although the JSR 168 standard assists in the adaptability of portlets to different portal applications it doesn't however define functionality desired of some portlets such as Inter Portlet Communication (IPC). In light of this some portal applications have created their own proprietary functionality (e.g. custom portlet modes and window states) which serves to enhance their own offering but likely inhibits portlet portability. The part of the Java community which develops the portlet standards are therefore currently working on the provision of a new standard (JSR 286) based on feedback and requests from the portal/portlet user/developer community which will contain guidance for new functionality.

3.3 Development Strategies

The IBVRE first introduced five portlets for communication with grid services which had been previously used in a Gridsphere portal application. To achieve this relocation certain Gridsphere-specific configurations were removed and/or modified to reflect the uPortal specifications, principally in the web.xml configuration file, but apart from this there was almost no change necessary in terms of portlet deployment and content provision. These portlet applications though were very simplistic in terms of portlet functionality, in that depending on the portlet mode, e.g. VIEW, EDIT, or HELP, individual portlets would only need to perform relevant processing and present an appropriate view. As such they would extend the abstract javax.portlet.GenericPortlet class and override the processAction(ActionRequest, ActionResponse), doView(RenderRequest, RenderResponse), doEdit(RenderRequest, RenderResponse) and doHelp(RenderRequest, RenderResponse) methods.

Whilst such means are suitable for trivial portlet applications they can quickly become difficult to enhance if the application processes become more advanced. Furthermore, generally the EDIT mode is intended for use as a portlet configuration/customization mode rather than part of a business process.

As part of later IBVRE development work a more complex portlet application which would allow users to parameterize and run heart model simulation code on grid/cluster infrastructure was specified. An early proposition had been that since a prototype standalone application with a Struts client interface was being developed as a proof-of-concept then so a Struts Bridge portlet (part of the Apache Portals project) could easily be developed. After some investigations into the suitability of this approach, and some development complications owing to the design of the prototype, this proposition was rejected in favour of developing a true JSR 168-compliant portlet which used the Spring framework and Hibernate, and in particular the Spring Portlet MVC architecture. In this way the portlet would be able to utilise all the present and future portlet standards functionality. Although the learning curve can be steep when understanding both how to use Spring and Hibernate and the differences between developing applications for web applications and portal applications, e.g. two-phase portlet life-cycle and portlet URL assignment, the resulting application is much improved in terms of adaptability, testing and maintainability.

3.4 +ves of portlet use.

a) Although the availability of third party portlets (e.g. calendars, bulletin boards) is increasing, albeit currently at a slow pace, this means that additional functionality can be provided to the portal's users by simply deploying the new portlet into the portal rather than relying on in-house resources.

b) The ability to aggregate content in portal screens means that many applications are visible to the user simultaneously. Owing to the separate render and action phases of
portlets this content, if it is not cached, may be refreshed without necessarily interacting with the portlet itself i.e. interaction with one portlet causing the portal to issue a render request to all other visible portlets.

3.5 —ves of portlet use.

a) When conforming to portlet standards, namely the current JSR 168 standard, the design considerations need to reflect the absence of functionality which could enhance portlet use. This compromises design optimisations (e.g. caching, limited authentication support, IPC) and limits the utilisation of technologies which perhaps have yet to have portlet implementations or may conflict with portal technologies. There is also no guarantee that a portlet will have a unique occurrence within a visible area nor that it will be assigned an expected space within such an area.

b) Ambiguity of use of portlet modes, and lack of enforcement to conform to those modes, means that they can be used inappropriately which may lead to confusion.

c) Limited CSS class for consistent look and feel. Although there is are portlet CSS naming definitions the list is incomplete and can conflict with alternative definitions such as that used in WSRP.

3.6 Future

The JSR 286 (Portlet 2.0 specification) - due around May 2007 - is intended to extend the JSR 168 specification by including at least the following functionality:

• File Downloading

• Adoption of WSRP 2.0

• Portlet coordination through IPC and Events

• Portlet filtering (equivalent of servlet filters)

• AJAX support

• Improved caching facilities