IBM

Development of standards-based grid portals, Part 2: JSR 168 grid portlets

Level: Intermediate

Xiaobo Yang (x.yang@dl.ac.uk), Software Developer, Consultant
Xiao Dong Wang (x.d.wang@dl.ac.uk), Senior Software Developer, Consultant
Robert Allan (r.j.allan@dl.ac.uk), Group Leader, Consultant

03 Apr 2007

Built on top of grid middleware, grid portals act as gateways to the grid because they smooth the learning curve of using grid. In the first of this three-part "Development of standards-based grid portals" series, we give an overview of grid portals, focusing on today's standards-based (JSR 168 and WSRP 1.0) second-generation grid portals. In this article, we develop three portlets to illustrate how a grid portal can be built using JSR 168-compliant portlets. And in Part 3, we discuss the application of Web Services for Remote Portlets (WSRP) and the future of grid portals.

Required skills

Before testing our example grid portlets, you should be familiar with the following technologies:

- Java™ development environment -- Sun JDK V1.5.0_10 is used; the code is compatible with JDK V1.4 and was compiled successfully using JDK V1.4.2_08
- Servlet/JSP development.
- Portal framework -- The uPortal V2.5.3 quickstart is used; other portal frameworks, including eXo platform V1.0, GridSphere V2.0.2, Liferay professional V3.6.1, and StringBeans V2.4.2, have been tested.
- Grid and Globus Toolkit -- GT V2.4 is used; if you are going to use GT V3.x/4.x, please change utility/JobSubmissionPortlet classes where appropriate.
    - Grid Security Infrastructure (GSI)
    - MyProxy
    - Globus Resource Allocation Manager (GRAM)
    - Metacomputing Directory Service (MDS)
    - Java CoG, cog-jglobus V1.4 is used

Java and servlet/JSP development skills are essential to write portlets. The LdapBrowser portlet doesn't require GSI for authentication and authorisation because MDS servers are normally open to public. For the grid examples, you need to have a good understanding of GSI, MyProxy, and GRAM. If you are going to test the ProxyManager and JobSubmission portlets, user and host certificates are required in addition to your CA root certificate(s). You need to set up all these parameters during Java CoG setup. Please make sure you are able to run your Java CoG command-line scripts before you deploy our grid portlets. Furthermore, you are required to have access to a MyProxy server and a GRAM gatekeeper. The examples given here were developed and deployed on Linux®, but should also run on Windows®. You will also notice the possible emerging firewall issues.

Writing grid portlets using Java CoG

JSR 168 portlet

There are plenty of tutorials (see Resources) to guide you through developing a JSR 168 portlet. Here, we give some general instructions without a detailed step-by-step guide.

PortletSession -- from JSR 168

- The PortletSession interface defines two scopes for storing objects:`APPLICATION_SCOPE`

Each portlet application is a Web application. One portlet application may have more than one portlet inside. This is important for our grid portlets because portlets within one portlet application can share an application-level session. When a user retrieves a proxy credential from a MyProxy server using the ProxyManager portlet, the JobSubmission portlet needs to use it so that a job can be submitted to a remote computing node on behalf of the user.

Here, we can put these two portlets inside one portlet application to share the proxy credential. In practice, the proxy credential can also be stored in a database. Thus, there's no need to put many portlets inside a single portlet application. In our example, however, putting all portlets inside one portlet application makes the code easy to read because database-related code is removed.

and `PORTLET_SCOPE`.

- Any object stored in the session using `APPLICATION_SCOPE` is available to any other portlet that belongs to the same portlet application and that handles a request identified as being a part of the same session.
- Objects stored in the session using `PORTLET_SCOPE` must be available to the portlet during requests for the same portlet window from which the objects were stored.

The first portlet we're going to develop is the LdapBrowser portlet, which is used to query LDAP/MDS servers to get information about grid resources within a virtual organisation (VO). Although there's no need to share a credential with the other two portlets, this portlet is put in the same portlet application called "grid," as well. The portlet itself contains the following source files and image files. Please be aware that here web.xml and portlet.xml are shared with the other two portlets. If you have a look at the source files, you'll find several web.xml files under directories like *exo* and *uportal*. This is because each portal framework has its own mechanism for loading a portlet application. Also remember that each portal framework has its own tag file to implement tags for the portlet library. In fact, the only work to be done when you deploy portlets in a different portal framework is to replace web.xml and the portlet tag file. Source and image files:

- src/uk/ac/clrc/escience/ngsportlets/ldapbrowser/LdapBrowserPortlet.java
- src/uk/ac/clrc/escience/grid/util/LdapNodeAttribute.java
- src/uk/ac/clrc/escience/grid/util/LdapNodeLink.java
- web-content/jsps/ldapbrowser-edit.jsp
- web-content/jsps/ldapbrowser-help.jsp
- web-content/jsps/ldapbrowser-view.jsp
- web-content/WEB-INF/web.xml
- web-content/WEB-INF/portlet.xml

A portlet follows the Model-View-Controller (MVC) design pattern. As you can see from the above list, the LdapBrowserPortlet portlet has three view pages -- ldapbrowser-edit.jsp, ldapbrowser-help.jsp, and ldapbrowser-view.jsp -- which represent the three *mode*s this portlet supports. Supported modes of a portlet are defined in the portlet application configuration file called portlet.xml. There are other settings, such as `supported-locale`, but most importantly, `portlet-preferences` sets the initial preferences of a portlet. A key feature of portlets is that different users can set different preference values for the same portlet. For example, three preference values are defined for the LdapBrowser portlet: `ldap_hostname`, `ldap_hostport`, and `ldap_basedn`. The edit mode of a portlet is designed for setting up users' own preferences.

The LdapBrowserPortlet class has the structure in Listing 1. The `init` method is used for some initial settings for a portlet. For example, in the LdapBrowser portlet, this method is used to load URLs of three jsp pages. Although not defined in Listing 1, a `destroy` method can be used to do some cleanup work like closing a database connection before the portlet dies. `processAction` is responsible for handling action requests from end users, while the three `doXxx` methods are related to different modes of this portlet. The core business logic resides in the `doGetldapinfo` method, which does an LDAP query with parameters collected from the user interface.

Listing 1. Structure of LdapBrowserPortlet class

```
public class LdapBrowserPortlet extends GenericPortlet {
  public void init(PortletConfig pConfig) throws PortletException {
  }

  public void processAction(ActionRequest request, ActionResponse response)
                                        throws PortletException, IOException {
  }
```

```
  public void doView(RenderRequest request, RenderResponse response)
                                        throws PortletException, IOException {
  }

  public void doEdit(RenderRequest request, RenderResponse response)
                                        throws PortletException, IOException {
  }

  public void doHelp(RenderRequest request, RenderResponse response)
                                        throws PortletException, IOException {
  }

  private void doGetLdapInfo(ActionRequest request, ActionResponse response)
                                        throws PortletException, IOException {
  }

}
```

Java CoG provides an implementation of Java-based GSI, GridFTP, MyProxy, GRAM client, etc. With the help of Java CoG, grid application developers can make use of high-level Java APIs to create and retrieve proxy credentials, submit jobs to remote GRAM gatekeepers, and perform file transfers using GridFTP and others. For this reason, Java CoG is the basis of today's GT-based grid portals.

Here, we are going to develop another two grid portlets: ProxyManager and JobSubmission. The former will be used to retrieve and update a proxy credential from a MyProxy server and save it in the session; the latter will be used to submit a job to a remote GRAM gatekeeper. The JobSubmission portlet will retrieve the proxy credential from the session because in our example, both portlets belong to the same portlet application.

If you have a look at the source code, you will find that we have a set of classes in the uk.ac.clrc.escience.grid.util package. For example, ProxyUtil and GRAMJob have core business logic implemented for the ProxyManager and JobSubmission portlets, respectively. This approach makes it easy for testing business logic. Moreover, the business logic can be published as Web services to permit maximum code reusability. Portlet developers can then focus on the presentation layer.

Compile, deploy, and test grid portlets

The source code of the LdapBrowser, ProxyManager, and JobSubmission portlets can be downloaded from the Download section. Once unzipped, enter the GridPortlet directory and read the readme.txt file to get instructions.

In general, before you compile the grid portlets, have a look at the build.properties file and download the required libraries, including Java CoG V1.4, JDOM V1.0, servlet and portlet APIs and put them under the lib directory. Also, set the portlet-container.home parameter within build.properties to point to Tomcat, where your portal framework is installed. Now you're ready to compile and deploy the code. Type `ant`, and you'll get all available options. For example, to compile and deploy the portlets under uPortal, use these commands:

- `ant build`
- `ant war.uportal`
- `ant deploy.uportal`

If everything goes well, you can start up your uPortal and publish these three portlets before adding them to a portal page. For more information about how to publish and deploy portlets inside uPortal, please refer to the uPortal user guide. When publishing a portlet, check the Portlet Definition ID setting in uPortal. This parameter is defined in web.xml as `portlet-guid` for each portlet. In our example, we define them as `grid.LdapBrowserPortlet`,

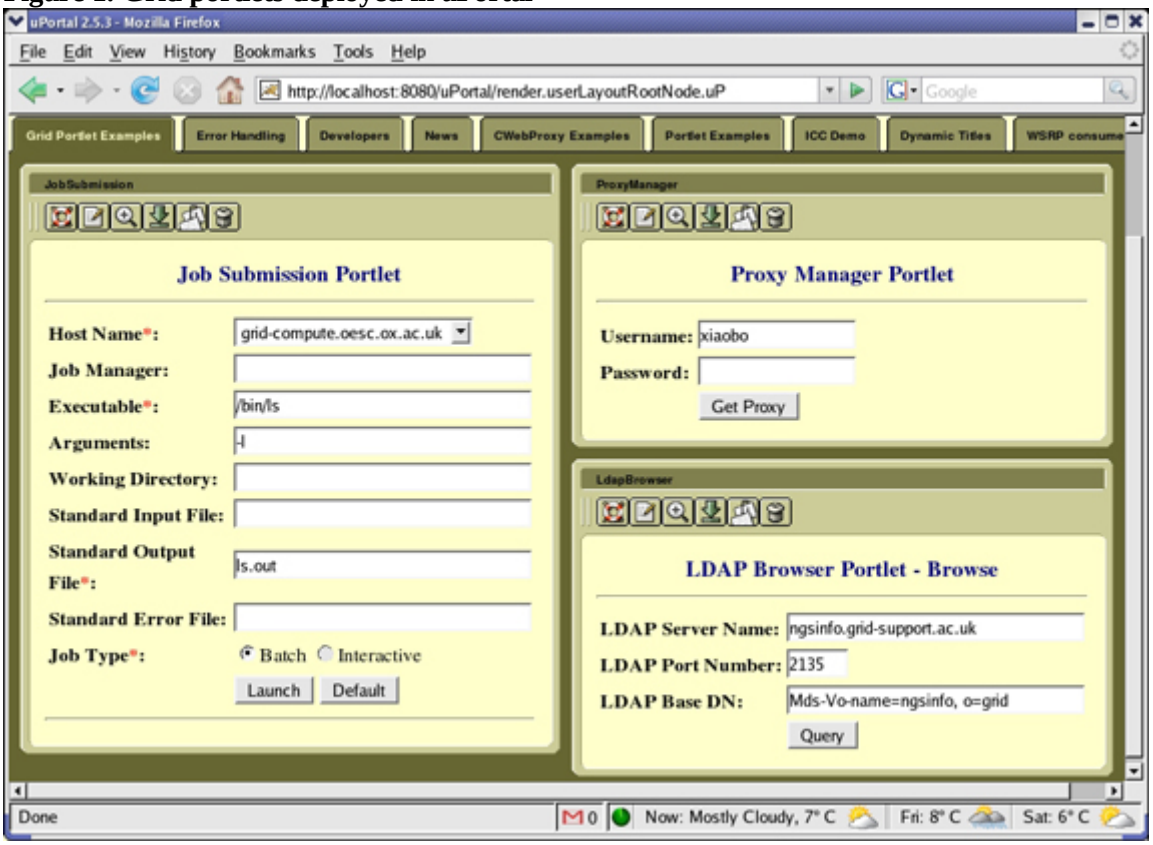
Portlet modes and window states

The JSR 168 specification defines three modes: view, edit, and help. The view mode is required to render markups of a portlet. The optional edit and help modes are defined to provide a user interface for customisation and for providing portlet help information. JSR 168 also allows the definition of vendor-specific modes.

JSR 168 also defines three window states -- *normal*, *maximized*, and *minimized* -- to indicate the amount of portal page space to be assigned to the content generated by a portlet. Similar to portlet modes, JSR 168 allows vendor-specific window states.

`grid.ProxyManagerPortlet`, and `grid.JobSubmissionPortlet`. Figure 1 shows a tab named Grid Portlet Examples with the three portlets we have developed. For the LdapBrowser portlet, the default user name has been changed from the default "tom" to "xiaobo" using edit mode.

Figure 1. Grid portlets deployed in uPortal



Before testing ProxyManager and JobSubmission, obtain a user certificate and host certificate from your CA. The user certificate is used to upload a long-term proxy credential to a MyProxy server from which a short-term proxy credential can be retrieved by the ProxyManager portlet. This is a widely used security procedure for grid portals. The host certificate is required by the ProxyManager so it can authenticate the server from which the proxy credential request comes. The host certificate and private key are hardcoded in uk.ac.clrc.escience.grid.util.PortalCredential as "/etc/grid-security/hostcert.pem" and "/etc/grid-security/hostkey.pem." If you store them in a different location, you can change them within the PortalCredential class or call the following:

```
public static GSSCredential getProxyCredential(String myproxy_host, int
myproxy_port, String myproxy_username, String myproxy_password, int
lifetime_in_second, String hostcert, String hostkey)
```

rather than the method below from ProxyUtil inside ProxyManagerPortlet:

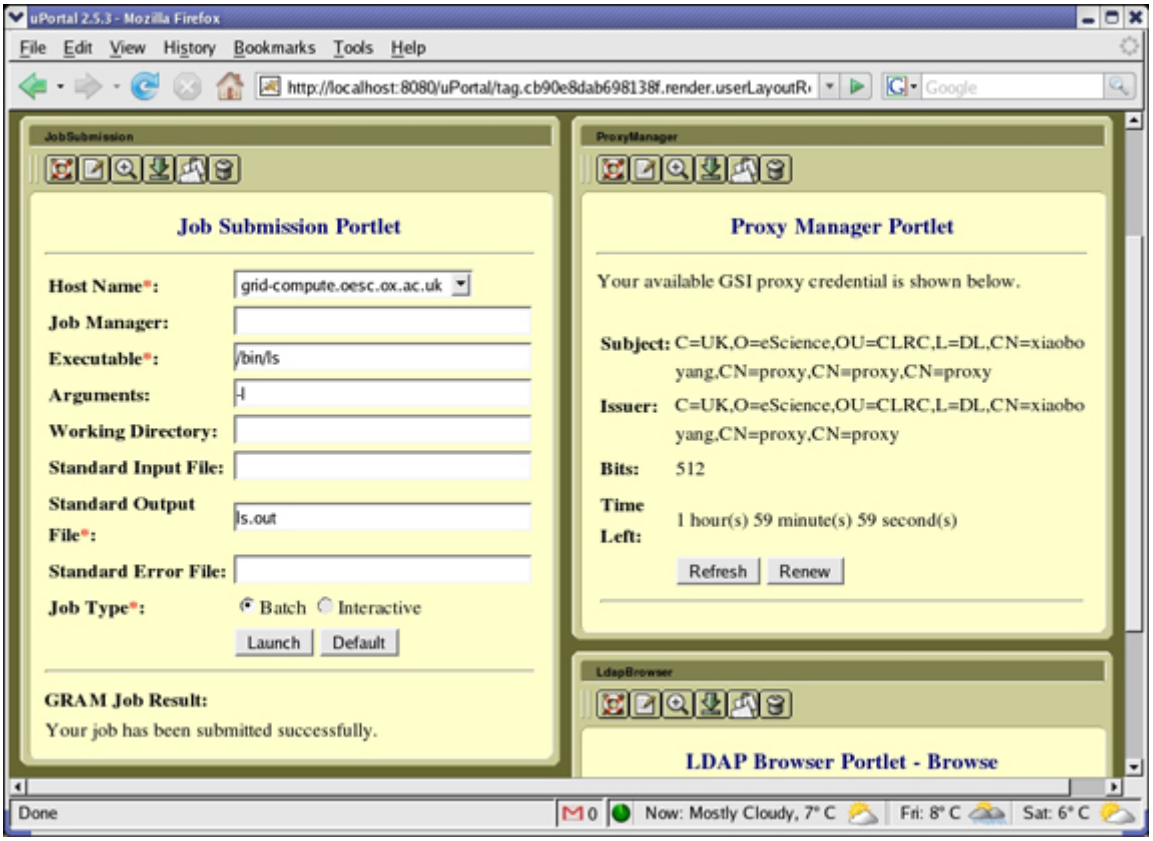
```
public static GSSCredential getProxyCredential(String myproxy_host, int
myproxy_port, String myproxy_username, String myproxy_password, int
lifetime_in_second) .
```

A final step before you test the ProxyManger and JobSubmission portlets is to upload a long-term proxy credential to your MyProxy server. This can be done using the following Java CoG command:

```
$COG_INSTALL_PATH/bin/myproxy -h MyProxy_server -p MyProxy_port put
```

Now you're ready to test both portlets. First use ProxyManager to retrieve a short-term proxy credential. Because this credential is stored in the session, the JobSubmission portlet will automatically pick it up so that a job can be submitted via a remote GRAM gatekeeper to a grid resource. Figure 2 displays information about a proxy credential and a job which has been submitted.

Figure 2. A job has been successfully submitted to a remote GRAM gatekeeper



The JobSubmission portlet also records the job information into an XML file called jobs.xml in the grid/WEB-INF directory. The job just submitted is recorded, as shown in Listing 2. In the production version of a grid portal, such information may be recorded in a database for future use. More functionalities are obviously required for building a useful grid portal. For example, GridFTP-based file transfer is essential. Develop your own grid portlets for your requirements and share them.

Listing 2. A job recorded in jobs.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ngsbatchjobs>
  <batchjob>
    <dn>C=UK,O=eScience,OU=CLRC,L=DL,CN=xiaobo yang,CN=proxy,CN=proxy,CN=proxy</dn>
    <date>1165582518358</date>
    <jobid>https://grid-compute.oesc.ox.ac.uk:64493/27245/1165582519/</jobid>
    <hostname>grid-compute.oesc.ox.ac.uk</hostname>
    <jobmanager />
    <executable>/bin/ls</executable>
    <arguments>-l</arguments>
    <directory />
    <stdin />
    <stdout>ls.out</stdout>
    <stderr />
  </batchjob>
</ngsbatchjobs>
```

Summary

Using the information presented in the grid portal review in Part 1 of this series, we developed three portlets: LdapBrowser for LDAP/MDS query, ProxyManager for retrieving and updating proxy credentials for end users, and JobSubmission for submitting jobs to a remote GRAM gatekeeper on behalf of those users. A productive grid portal normally allows MyProxy logon for authentication to the portal itself. This

involves modifying a portal framework like StringBeans, which we did in release 2 of the NGS Portal. Auto-mapping of a MyProxy authenticated user to a local portal user is required so that all portal functionalities are supported.

In Part 3 of this "Development of standards-based grid portals" series, we talk about our WSRP experiences and the future of grid portals.

Download

Description	Name	Size	Download method
Source code	gr-stdsportal2.GridPortlet.zip	2MB	**HTTP**

→ Information about download methods

Resources

Learn

- Read Part 1 of this "Development of standards-based grid portals" series.
- JSR 168 is a portlet specification developed by JCP to standardise communications between a portlet and a portlet container.
- Read the Introducing Java Portlet Specifications: JSR 168 and JSR 286 from Sun Developer Network.
- Read the developerWorks article "Best practices: Developing portlets using JSR 168 and WebSphere Portal."
- Grid Security Infrastructure (GSI) is the basis of Globus Toolkit to provide authentication, secure communication, single sign-on, and more for grid users.
- Grid Resource Allocation and Management (GRAM) provides a single interface for requesting and using remote system resource for the executing jobs.
- GridFTP is a high-performance, secure, reliable data-transfer protocol optimised for high-bandwidth wide-area networks.
- Monitoring and Discovery Service (MDS) is the information services component of the Globus Toolkit and provides information about the available resources on the grid and their status.

Get products and technologies

- The Globus Toolkit is the de-facto standard implementation for enabling the grid.
- Java Commodity Grid (CoG) Kits allow grid users, application developers, and administrators to use, program, and administer grids from a higher-level framework. It provides implementation of Java-based GSI, GridFTP, MyProxy, GRAM client, and additional functionalities.
- MyProxy is open source software from NCSA for managing X.509 Public Key Infrastructure (PKI) security credentials (certificates and private keys).
- Check out eXo Portal, which helps you build integrated solutions with a secure infrastructure, essential for developing an effective portal.

- GridSphere is a portal framework that provides an open source portlet-based Web portal.
- Liferay is an enterprise open source portal framework.
- uPortal is a free, sharable open source portal under development by higher educational institutions.

About the authors

Xiaobo Yang is a software developer working in the Grid Technology Group, CCLRC e-Science Centre, in the United Kingdom. He is interested in grid, collaborative virtual research environments, and various Web technologies, including grid portals, and Service-Oriented Architecture (SOA).

Xiao Dong Wang is a senior software developer in the Grid Technology Group, CCLRC e-Science Centre, in the United Kingdom. His interests include grid middleware design and application, portal user interfaces and portlet development, JSP, and JSF. He has more than six years' experience in Java programming and Web and grid service development.

Robert Allan leads the Grid Technology Group, CCLRC e-Science Centre, in the United Kingdom. His background is as a physicist and -- since the mid-1980s -- developer of high-performance computing applications using the latest technologies. He has managed several large HPC and e-science projects in the United Kingdom. He is particularly interested in making the grid widely usable.