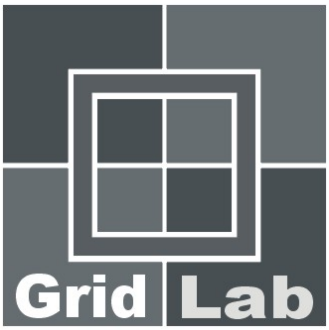


Jetspeed Evaluation

Author(s):	Ian Kelley, Jason Novotny, Michael Russell, Oliver Wehrens
Document Filename:	
Work package:	WP 4 - Portals
Partner(s):	
Lead Partner:	AEI Potsdam
Config ID:	
Document classification:	

Abstract: This internal document evaluates the Jetspeed project as a viable platform for developing portlets for use within the GridLab portal.

Contents

1 Introduction

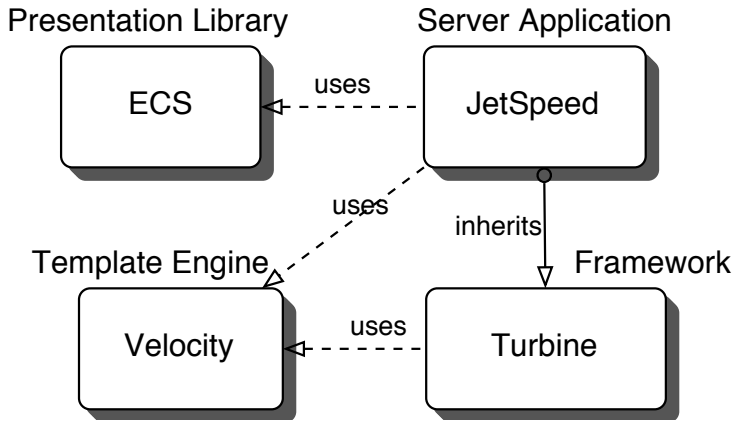
Portlets [1] have become an increasingly popular concept used to describe a visual user interfaces that provides some amount of customization and personalization capabilities and are traditionally used to provide content from information providers e.g. ApacheWeek [3] or Slashdot [2]. Portlets are also used to provide access to applications or services. Multiple portlets can be aggragated on s ingle portal page providing users with easy access to a range of services/applications suitable for a particular problem domain. In the world of Grid computing, portlets have been growing in popularity and the GCE WG of Grid Forum has been prototyping portlet based access to Grid services which may also include services supported under the OGSA framework [4]. The Portlet API exists as a Java Specification request (JSR) and is in the process of being standardized in the Java Community Process (JCP).

2 Jetspeed

The Jetspeed [5] project led by the open source Apache Jakarta foundation has provided an implementation of the Portlet API and a portal development kit designed to ease the development of new portlets. A portal user interface is composed of multiple portlets that each provide access to a particular service or set of services as well as providing content from information providers. Portlets leverage XML and XSL technologies to provide support for multiple devices using a different set of tags to display information e.g. WML, RSS. The Jetspeed project supports the following features listed off the website:

- Soon to standardize on a Java Portlet API specification
- Template-based layouts including JSP and Velocity
- Supports remote XML content feeds via Open Content Syndication
- Custom default home page configuration
- Database user authentication
- In-memory cache for quick page rendering
- Rich Site Summary support for syndicated content
- Integration with Cocoon, WebMacro and Velocity so that you can develop with the newest XML/XSL technology.
- Wireless Markup Language (WML) support
- XML based configuration registry of portlets
- Web Application development infrastructure
- Local caching of remote content
- Synchronization with Avantgo
- Integrated with Turbine modules and services
- Profiler Service to access portal pages based on user, security (groups, roles, acls), media types, and language

- Persistence Service available to all portlets to easily store state per user, page and portlet
- Skins so that users can choose colors and display attributes
- Customizer for selecting portlets and defining layouts for individual pages
- PSML can be stored in a Database.
- User, group, role and permission administration via Jetspeed security portlets.
- Role-based security access to portlets.



3 Turbine

Turbine provides a model-view-controller (MVC) framework for constructing web application. In a standard MVC or Model 2 framework, [11] the presentation of a web application is separated from the so-called business or back-end logic using a combination of servlets and JSP. A servlet acts as a controller to handle browser requests and instantiate Java beans that may be used to access back-end resources such as databases, Grid resources, etc or create ”view” beans that encapsulate the resulting data to be a displayed in the browser. The servlet also performs the processing logic to determine the JSP page to forward control to by using special tags as part of the client's request and a table that maps the tag to the appropriate JSP page . Finally the JSP page merely presents a suitable display of the data in the ”view” bean.
The MVC architecture adopted by Turbine attempts to provide more flexibility in the view by supporting other page markup languages besides JSP and HTML pages. The display pages may consist of more complex template systems such as WebMacro, Velocity or FreeMarker. Velocity has gained the most acceptance within the Jetspeed community and will be discussed in the next section. The controller, the TurbineServlet can make use of a multitude of Turbine services including the following list:

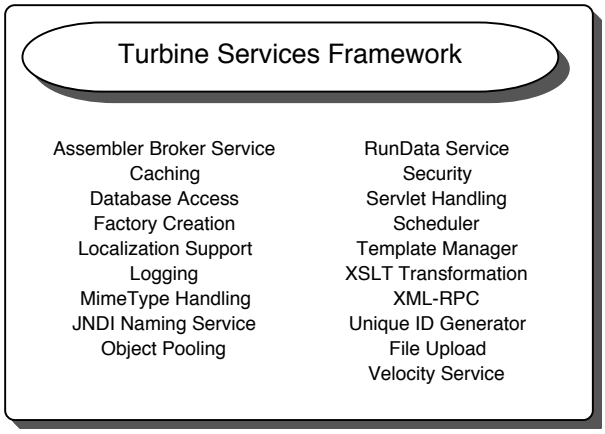


Figure 1: Provided Turbine Services

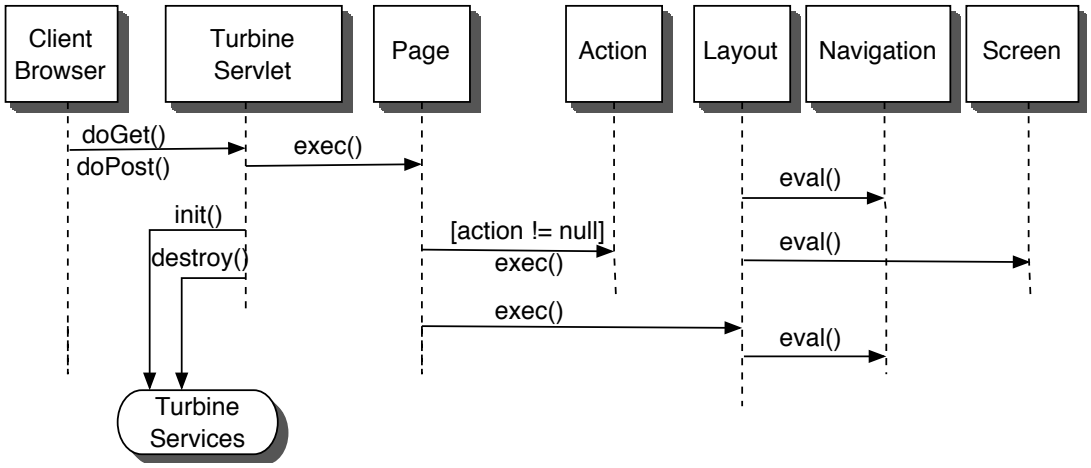


Figure 2: Turbine Sequence Diagram

Services are defined as singleton classes and instantiated and destroyed by the TurbineServlet. A Turbine properties file is maintained to allow the dynamic setting of various service attributes e.g. max file size for the file upload service or caching parameters, etc. Consequently, the properties file has grown quite large as the number of services and service options grows. In the next version of Turbine, version 3, many of the services are packaged separately under the Fulcrum library which is currently under development.

The flow of events in the Turbine framework is outlined in the following sequence diagram.

The Turbine MVC framework is comprised of the following key modules: Page, Action, Layout, Navigation and Screen objects. The Turbine servlet initializes the services listed in its properties file and then forwards control to a Page object which is responsible for executing the Layout object and invoking an Action object if one was defined. The Layout object is responsible for executing the navigation and screen objects which defines the web site navigation as well as the physical layout of the web page. Visually, the framework modules map to the physical web page as shown below.

The Jakarta foundation also supports another competing Model 2 framework, Struts, which has been growing in popularity and acceptance. Primary differences between the two appear to be

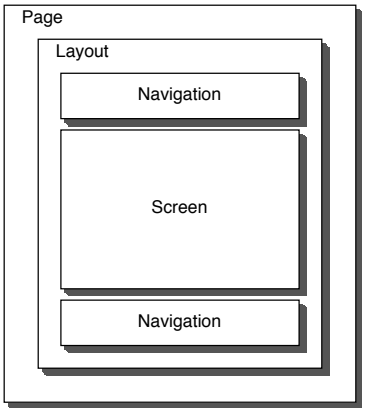


Figure 3: Turbine Modules Representing a Template Web Page

support for the Velocity template engine and more bundled ”services” such as servlet handled security, object caching, SQL database access and a default user profile.

4 Velocity

Velocity is a general purpose Java-based template engine. Velocity provides an alternative scripting language to JSP for developing portal pages.
The following is a sample page comparison between using JSP and Velocity to present the canonical ”Hello, World” in a web browser:
JSP:

```
<html>
<head><title>Hello</title></head>
<body>
<h1>
<%
if (request.getParameter("name") == null) {
   out.println("Hello World");
} else {
  out.println("Hello, " + request.getParameter("name"));
}
%>
</h1>
</body></html>
```

Velocity:

```
<html>
<head><title>Hello</title></head>
<body>
<h1>
#if ($request.getParameter("name") == null)
   Hello World
#else
```

```
   Hello, $request.getParameter("name")
#end
</h1>
</body></html>
```

5 ECS

The Element Construction Set [?] is a Java API for generating elements for various markup languages including HTML 4.0 and XML. ECS contains Java classes for the representation of HTML or XML tags. An example of creating an HTML page using ECS is shown below:

```
Html html = new Html()
               .addElement(new Head()
                   .addElement(new Title("Demo")))
               .addElement(new Body()
               .addElement(new H1("Demo Header"))
               .addElement(new H3("Sub Header:"))
               .addElement(new Font().setSize("+1")
                           .setColor(HtmlColor.WHITE)
                           .setFace("Times")
                           .addElement("The big dog \& the little cat chased each other.")));
out.println(html.toString());
// or write to the outputstream directly
output(out);
```

Although ECS is useful to provide support for multiple mark-up tags, it still requires that mark-up information be placed in compiled code. For this reason, ECS will be deprecated in the 2.0 release of Jetspeed.

6 Portlet Development Using Jetspeed

Below is a canonical example of the ”Hello World” portlet using Jetspeed:

```
package com.bluesunrise.portal.portlets;

import org.apache.jetspeed.portal.portlets.AbstractPortlet;
import org.apache.turbine.util.RunData;
import org.apache.ecs.ConcreteElement;
import org.apache.ecs.StringElement;

public class HelloWorldPortlet extends AbstractPortlet
{
    public ConcreteElement getContent (RunData runData)
    {
        return (new StringElement ("Hello World!"));
    }
}
```

In Jetspeed, all portlets inherit from AbstractPortlet and must provide a getContent method to supply the portlet information. This example uses StringElement, an ECS class, that can translate to WML or HTML as is appropiate. The RunData object that gets passed in to getContent is a Turbine object that stores various Servlet objects .e.g ServletContext, ServletRequest, HttpSession, etc as well as other Turbine objects including Users.
The next step required in deploying a portlet is to add the portlet to the Jetspeed Portlet Registry. A registry fragment file must be created that provides the portlet name, a description and the classname. The Hello World example registry file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<registry>
    <portlet-entry name="HelloWorld" hidden="false" type="instance" application="false">
        <meta-info>
            <title>HelloWorld</title>
            <description>Portlet How To Example 1 ■ Hello World</description>
        </meta-info>
        <classname>com.bluesunrise.portal.portlets.HelloWorldPortlet</classname>
        <media-type ref="html"/>
    </portlet-entry>
</registry>
```

7 IBM's Portlet API Specification

IBM has been one of the early supporters of the Portlet API JSR. IBM WebSphere Portal Server version 1.2 shipped with JetSpeed as the portlet implementation. The latest version, 4.1, as of this writing includes a new specification and implementation of the Portlet API [8] that looks to be closed source for now. Several well written documents from IBM describe migration issues from the Jetspeed Portlet API to the current API [9]. The design of the portlet API released in WebSphere 4.1 closely mimicks the Servlet API [**?**] and Portlet classes directly inherit from their Servlet counterparts. For example, PortletRequest extends ServletRequest, etc. Judging from the supplied documentation on developing portlets [7] and the draft specification of the portlet API [8], the WebSphere 4.1 Portlet API appears to be much more clear and well defined than the specification implemented in Jetspeed. Possibly, the two projects may converge when the Portlet API JSR is slated to become approved by the end of the year.
Below is a canonical example of the "Hello World" portlet using IBM WebSphere 4.1:

```
package com.ibm.wps.samplets.helloworld;
import org.apache.jetspeed.portlet.*;
import org.apache.jetspeed.portlets.*;
import java.io.*;

public class HelloWorld extends PortletAdaptor
{
   public void init(PortletConfig portletConfig) throws UnavailableException
   {
      super.init( portletConfig );
   }

   public void service(PortletRequest portletRequest,
```

```
                        PortletResponse portletResponse) throws PortletException,
                                                               IOException
   {
         PrintWriter writer = portletResponse.getWriter();
         writer.println("<p>Hello Portal World!</p>");
   }
}
```

Unlike the Jetspeed example, there are no references to Turbine objects using WebSphere. In addition, the portlet extends from PortletAdaptor rather than AbstractPortlet and offers an API much more similar to the Servlet API. Resources are initialized in the init method and the service method is used to handle client requests. More information on differences between the Jetspeed API and WebSphere's Portlet API can be found in the Portlet Migration Guide [9] available from IBM.
Below is the portlet registry file used for the IBM WebSphere example, which offers many more options over the Jetspeed portlet registry:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE portlet-app-def PUBLIC "-//IBM//DTD Portlet Application 1.1//EN"
"portlet_1.1.dtd">
<portlet-app-def>
<portlet-app uid="com.ibm.samples.HelloWorld.4943.1"> <portlet-app-name>Hello World Portlet
<portlet href="WEB-INF/web.xml#Servlet_439329280" id="Portlet_439329280">
<portlet-name>HelloWorld</portlet-name>
<cache>
<expires>0</expires>
<shared>no</shared>
</cache>
<allows> <minimized/> </allows>
<supports>
<markup name="html">
<view/>
</markup>
</supports>
</portlet>
</portlet-app>
<concrete-portlet-app uid="640682430">
<portlet-app-name>Concrete Hello World - Portlet Sample #1</portlet-app-name>
<context-param>
<param-name>Portlet Master</param-name>
<param-value>yourid@yourdomnain.com</param-value>
</context-param>
<concrete-portlet href="#Portlet_439329280">
<portlet-name>HelloWorld</portlet-name>
<default-locale>en</default-locale>
<language locale="en_US">
<title>Hello World - Sample Portlet #1</title>
<title-short>Hello-World</title-short>
<description>Hello World - Sample Portlet #1</description>
```

```
<keywords>portlet hello world</keywords>
</language>
</concrete-portlet>
</concrete-portlet-app>
</portlet-app-def>
```

8 GridLab Portal Considerations

The development of the GridLab portal, GridSphere, must adhere to the requirements put forth in the initial WP 4 Portals requirements document [6]. All of the requirements outlined may be met by using the Turbine/Jetspeed SDK's. One issue that has come up is the dependence on Netscape 6 or greater for use with Jetspeed, although this issue has been addressed by another group using Jetspeed and a workaround has been developed already. Other issues and requirements are raised when considering portlets for Grid computing and are discussed below:

8.1 Security

Access to Jetspeed portlets is granted by first providing a username and password that matches a username and password stored on a back-end database. Jetspeed (through Turbine) also provides the ability to create a new account with a user supplied name and password if one does not exist. Within portlets, Access Control Lists (ACL's) may provide refined permission based access by distinguishing between users, groups, roles and permissions. A user may have multiple roles within a group e.g. administrator and user.
Within the GridLab portal, access is granted by users who can access a GSI credential from a MyProxy credential repository. The credential is used to securely access Grid services and provides single sign-on to resources. In the existing ASCPortal, users are granted access provided they can retrieve a credential from a MyProxy repository. To support this, both Jetspeed and the core Turbine servlet controller would need to be considerably modified.

8.2 User profiles

In the Jetspeed model, user profiles are provided as TurbineUser objects. A TurbineUser stores the following fields: first name, last name, last login, username and password. Turbine also supports storing TurbineUsers in the form of a schema to a back-end database.
The GridLab portal would presumably extend the TurbineUser object to include support for other user information including job history, application information, preferences and other data. Turbine also tightly couples the information in the Turbine user profile data object with a back-end database. Presumbaly, much of the code defining database tables would need to be augmented to support the various data fields needed by the GridLab portal.

8.3 Access to Grid/OGSA services

Access to OGSA based services or web services in general requires a portlet that can be extended to support dynamic object creation from WSDL interfaces and use SOAP as a transport protocol. It doesn't appear that Jetspeed or Turbine provides any support for these technologies, so they must be added in.

9 Final Recommendation

Although development has progressed for over two years, the Jetspeed API continues to evolve and remains a fast moving target at the time of this writing. When considering the viability of Jetspeed as a portal development platform, careful consideration must be given to all of the dependent source packages that it relies upon. In the case of ECS, the presentation library, it is envisioned that it can be swapped out for any other similar presentation library or JSP, XML/XSLT. In the case of Turbine, the strong coupling becomes more problematic as the needs and requirements of Jetspeed must always be satisfied by the evolution of Turbine. As of this writing, the next release of Turbine has branched into three separate projects. Turbine v3 is used for the presentation layer and contains the main servlet, Torque is used for database connection/schema handling and Fulcrum is a services framework that provides a bundle of services similar to those in the currently used version of Turbine. The ideal solution would be to design a framework from the ground up, essentially implementing much of the functionality already included in Turbine and Jetspeed with a more rigid focus on developing portals and portlets specifically for Grid computing and accessing Grid services. This approach, however, has the disadvantage that a lot of development time is required before any services or interfaces to services can be developed within the portal. One of the short term goals of WP 4 is to have a working portal available that can be used and tested by end users. The development of the GridLab portal and the portlet framework should progress in two parallel tracks. First, the ASCPortal codebase is migrated to use the Jetspeed and Turbine frameworks. Development along this direction will require refactoring existing ASCPortal functionality and capabilities into portlets that can be used within Jetspeed. In parallel, it is our recommendation to develop a cleaner, more focused MVC and portlet framework based on the IBM specifications. Since development can proceed in parallel on both tracks, the final task will be to migrate services and portlets relying on Turbine and Jetspeed to the new proposed framework once it is complete.

References

[1] IBM What is a portlet?
http://www-3.ibm.com/software/webservers/portal/portlet.html [22 May 2002].

[2] The Slashdot Website
http://www.slashdot.com [22 May 2002].

[3] The Apache Week Website
http://www.apacheweek.com [22 May 2002].

[4] I. Foster, C. Kesselman, J. Nick, S. Tuecke The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. January, 2002.

[5] The Jakarta Jetspeed Project
http://jakarta.apache.org/jetspeed [22 May 2002].

[6] Kelley I, Novotny J, Russell M, Wehrens O. WP 4: Portals Requirements Document *Internal GridLab WP document* April 2002;

[7] Hesmer S., Fischer P., Buckner T., Schuster I. *Portlet Development Guide* April 2, 2002;

[8] Hesmer S., Hepper S., Schaeck T. *Portlet API (First Draft)* April 2, 2002;

[9] Java Servlet API
http://java.sun.com/product/servlet

[10] Lection David B. *WebSphere Portal Version 4.1: Portlet Migration Guide* April 22, 2002;

[11] Element Construction Set
http://jakarta.apache.org/ecs [22 May 2002].

[12] The JavaServer Pages Model 2 Architecture
http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html [12 Dec 1999]