

SFA Modernization Partner

United States Department of Education

Student Financial Assistance



**Task Order # 48, Enterprise Portal Strategy and
Product Selection – Phase II (Mod. 01)**

Portal Software Architecture Design

Task Order #48

Deliverable # 48.2.1

August 8, 2001



Table of Contents

1	EXECUTIVE SUMMARY	3
1.1	INTRODUCTION	3
1.2	APPROACH	3
2	PORTAL SOFTWARE ARCHITECTURE DIAGRAM	4
3	PRESENTATION LAYER OVERVIEW	5
3.1	APACHE STRUTS	5
3.1.1	<i>Functional Definition</i>	5
3.1.2	<i>External Interfaces</i>	6
3.1.3	<i>Content Residence</i>	6
3.1.4	<i>Server Location</i>	6
4	APPLICATION LAYER OVERVIEW	7
4.1	REGISTRATION PORTLET	7
4.1.1	<i>Functional Definition</i>	7
4.1.2	<i>External Interfaces</i>	7
4.1.3	<i>Content Residence</i>	7
4.1.4	<i>Server Location</i>	7
4.2	LOGIN PORTLET	7
4.2.1	<i>Functional Definition</i>	7
4.2.2	<i>External Interfaces</i>	7
4.2.3	<i>Content Residence</i>	8
4.2.4	<i>Server Location</i>	8
4.3	PERSONALIZATION PORTLET	8
4.3.1	<i>Functional Definition</i>	8
4.3.2	<i>External Interfaces</i>	8
4.3.3	<i>Content Residence</i>	8
4.3.4	<i>Server Location</i>	8
4.4	SEARCH PORTLET	9
4.4.1	<i>Functional Definition</i>	9
4.4.2	<i>External Interfaces</i>	9
4.4.3	<i>Content Residence</i>	9
4.4.4	<i>Server Location</i>	9
4.5	CALENDAR PORTLET	9
4.5.1	<i>Functional Definition</i>	9
4.5.2	<i>External Interfaces</i>	9
4.5.3	<i>Content Residence</i>	9
4.5.4	<i>Server Location</i>	10
4.6	FEEDBACK PORTLET	10
4.6.1	<i>Functional Definition</i>	10



4.6.2	<i>External Interfaces</i>	10
4.6.3	<i>Content Residence</i>	10
4.6.4	<i>Server Location</i>	10
4.7	HEADLINES PORTLET	10
4.7.1	<i>Functional Definition</i>	10
4.7.2	<i>External Interfaces</i>	10
4.7.3	<i>Content Residence</i>	11
4.7.4	<i>Server Location</i>	11
5	RCS (REUSABLE COMMON SERVICES) OVERVIEW	11



1 Executive Summary

1.1 Introduction

Most people first encountered the term “portal” when they visited Web gateways such as Yahoo or Lycos. Since that time, the word portal has been applied to any Web site of great significance. Consequently, there are as many definitions of portal as there are developers of Web sites. To eliminate misinterpretation, portals provide a secure, single point of interaction with diverse information, business processes, and people, personalized to a user’s needs and responsibilities.

Some industry analysts have coalesced around the concept of Horizontal and Vertical Portals. Horizontal portals are the primary infrastructure upon which a portal is built. This infrastructure includes the Presentation Layer, which are the Web user interface and a controller framework for easily utilizing software modules and services. Vertical Portals are instances of portals covering a specific domain or functional area along with its corresponding data. These portals include Registration, Login, Personalization, Search, Calendar, Feedback, and Headlines.

1.2 Approach

The most popular approach to create a portal architecture uses a commercial off the shelf portal framework. This approach has several advantages including the ability to use commercial portlets that can dramatically reduce development costs and improve portal functionality. Commercial portals also provide portlet administration, personalization, customization, content syndication, and integrated search capabilities. The disadvantage is the initial cost of the portal software and the need to still do portal development if commercial portlets are not available.

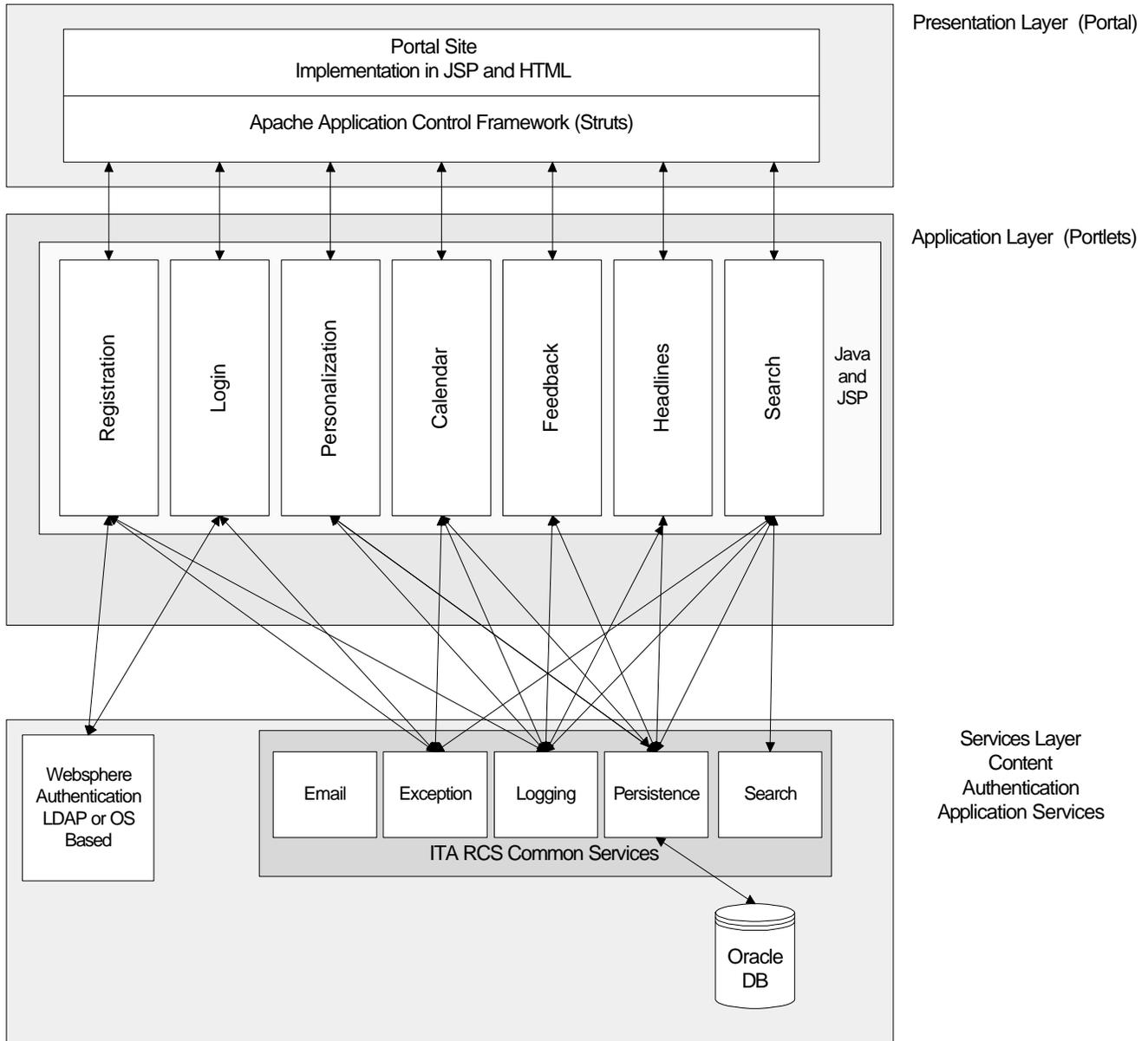
The other recommended approach is to create a lightweight portal framework. The benefit of this approach is initial cost. In designing a lightweight portal framework, the 3 primary architectural motivations are reduce the amount of custom code needed to create the framework by relying on open source frameworks as much as possible, reduce the development complexity of the portal, and ensure that portlets can be reused on multiple portal pages. The disadvantage is not using commercially available portlets and the potential for greater development costs.

A middle approach is not a viable option. It is not currently financially feasible to create a custom portal that can use off the shelf portlet components. It would be much cheaper to buy the framework than build it. It is also important to not over engineer the portal. Reusability of services should be of only secondary concern. Creating robust common services will quickly increase the cost of the project to the point that buying a portal framework would be attractive. Common services should be used if they are available via open source or they are critical to portal operation. Common services built from scratch that are not immediately needed should be grown organically through code refactoring and not architected in from the start.

The approach of creating a lightweight portal framework has been chosen for this task.



2 Portal Software Architecture Diagram





3 Presentation Layer Overview

3.1 Apache Struts

3.1.1 Functional Definition

From the Struts User's Guide,

"Struts applications have three major components: a servlet controller, Java Server pages (the "view"), and the application's business logic (or the "model"). Let's step through how this all fits together.

The controller bundles and routes HTTP requests to other objects in the framework, including Java Server Pages. When initialized, the controller parses a configuration resource file. The configuration resource defines (among other things) the action mappings for the application. The controller uses these mappings to turn HTTP requests into application actions.

At a minimum, a mapping must specify (1) a request path and (2) the object type to act upon the request. The action object can handle the request and respond to the client (usually a Web browser), or indicate that control should be forwarded to another action. For example, if a login succeeds, a loginAction object may wish to forward control to a main Menu action.

Action objects are linked to the application's controller, and so have access to that servlet's methods. When forwarding control, an object can indirectly forward one or more shared objects, including JavaBeans, by placing them in one of the standard collections shared by Java servlets.

An action object can create a shopping cart bean, add an item to the cart, place the bean in the session collection, and then forward control to another action -- that may use a JSP to display the contents of the user's cart. Since each client has their own session, they will each also have their own shopping cart. In a Struts application, most of the business logic can be represented using JavaBeans.

JavaBeans can also be used to manage input forms. A key problem in designing Web applications is retaining and validating what a user has entered between requests. With Struts, you can easily store the data for an input form in a form bean. The bean is saved in one of the standard, shared context collections, so that other objects, especially the action object, can use it.

The form bean can be used by a JSP to collect data from the user ... by an action object to validate the user entered ... and then by the JSP again to re-populate the form fields. In the case of validation errors, Struts has a shared mechanism for raising and displaying error messages.

A Struts form bean is defined in the configuration resource and linked to an action mapping using a common property name. When a request calls for an action that uses a form bean, the controller servlet either retrieves or creates the form bean, and passes it to the action object. The



action object can then check the contents of the form bean before its input form is displayed, and also queue messages to be handled by the form. When ready, the action object can return control with a forwarding to its input form, usually a JSP. The controller can then respond to the HTTP request and direct the client to the Java Server Page.

The Struts framework includes custom tags that can automatically populate fields from a form bean. The only thing most Java Server Pages need to know about the rest of the framework is the proper field names and where to submit the form. Components like the messages set by the action object can be output using a single custom tag. Other application-specific tags can also be defined to hide implementation details from the JSPs.

The custom tags in the Struts framework are designed to use the internationalization features built into the Java platform. All the field labels and messages can be retrieved from a message resource, and Java can automatically provide the correct resource for a client's country and language. To provide messages for another language, simply add another resource file. “

Struts are part of the [Jakarta Project](#), sponsored by the [Apache Software Foundation](#). The official Struts home page is at <http://jakarta.apache.org/struts>.

3.1.2 External Interfaces

Apache Struts is a Servlet and a set of helper classes along with Java Tag Libraries. The API's for Struts can be found at:

<http://jakarta.apache.org/struts/api/index.html>

3.1.3 Content Residence

Content for struts is confined to properties files that are deployed on the Portal's WebSphere Application Server.

3.1.4 Server Location

Apache Struts is deployed on the Portal's WebSphere Application Server.



4 Application Layer Overview

4.1 Registration Portlet

4.1.1 Functional Definition

The Registration Portlet provides the ability to register to the Schools Portal site. User will provide the input requested by the current registration service and will receive a user ID and password. The userid and Password are returned in real-time to the user on the screen. A user account will be created within an LDAP directory.

4.1.2 External Interfaces

The Registration Portlet like all portlets provides no unique interface of its own. Because portlets are implemented as Java Servlets they follow the Java Servlet API. The Registration Portlet can be called from a Template JSP via the include file tag since it is statically rendered:

```
<jsp:include file="/RegistrationPortlet" />
```

4.1.3 Content Residence

Registration Content will be stored in an LDAP directory. Registration data will be written via the standard Java JNDI API.

4.1.4 Server Location

Like all portlets, the Registration Portlet will be deployed on the Portal's WebSphere Application Server.

4.2 Login Portlet

4.2.1 Functional Definition

The Login Portlet performs login and authentication of a registered userid. This portlet will utilize existing WebSphere authentication API's. As WebSphere requires an LDAP directory, JNDI calls to an LDAP directory will be made.

4.2.2 External Interfaces

The Login Portlet like all portlets provides no unique interface of its own. Because portlets are implemented as Java Servlets they follow the Java Servlet API. The Login Portlet can be called from a Template JSP via the include file tag since it is statically rendered:

```
<jsp:include file="/LoginPortlet" />
```



4.2.3 Content Residence

Login Content will be stored in an LDAP directory. WebSphere Application Server will perform authentication with the LDAP directory via the JNDI API.

4.2.4 Server Location

Like all portlets, the Login Portlet will live on the Portal's WebSphere Application Server.

4.3 Personalization Portlet

4.3.1 Functional Definition

1. The Personalization Portlet provides the capability for a registered, logged on user, to personalize their home page. Once logged into the site a users default Home Page will appear. The first time a user registers and logs on to the site their home page is identical to the Schools Portal Main Home Page. The user should then have the capability to personalize the following:

- Change password – Note: this feature does not include any password management capabilities
- Add/Modify weblinks

4.3.2 External Interfaces

The Personalization Portlet like all portlets provides no unique interface of its own. Because portlets are implemented as Java Servlets they follow the Java Servlet API. The Personalization Portlet can be called from a Template JSP via the include page tag since it is dynamically rendered:

```
<jsp:include page="/PersonalizationPortlet" />
```

4.3.3 Content Residence

Personalized links will live on an Oracle Database. Data will be read and written via the standard Java JDBC API or via the ITA Persistence API if it proves suitable for the Personalization Portlet. Password changes will utilize JDBC to query and write to the LDAP directory.

4.3.4 Server Location

Like all portlets, the Personalization Portlet will be deployed on the Portal's WebSphere Application Server.



4.4 Search Portlet

4.4.1 Functional Definition

The Search Portlet provides an interface to the Autonomy search engine. Currently this interface is implemented using CGI. The CGI functionality will be removed and an API utilized.

4.4.2 External Interfaces

The Search Portlet like all portlets provides no unique interface of its own. Because portlets are implemented as Java Servlets they follow the Java Servlet API. The Search Portlet can be called from a Template JSP via the include file tag since it is statically rendered:

```
<jsp:include file="/SearchPortlet" />
```

4.4.3 Content Residence

Content for the Search Portlet will live in the Autonomy search engine. Communication with the Autonomy Search engine will occur via the standard Autonomy API's

4.4.4 Server Location

Like all portlets, the Search Portlet will be deployed on the Portal's WebSphere Application Server.

4.5 Calendar Portlet

4.5.1 Functional Definition

The Calendar Portlet will add a static calendar to display on the Schools Portal Home Page or on a logged on users Home Page. The calendar data will be static and will not contain user calendar specific information.

4.5.2 External Interfaces

The Calendar Portlet like all portlets provides no unique interface of its own. Because portlets are implemented as Java Servlets they follow the Java Servlet API. The Calendar Portlet can be called from a Template JSP via the include page tag since it is dynamically rendered:

```
<jsp:include page="/CalendarPortlet" />
```

4.5.3 Content Residence

Calendar entries will live on an Oracle Database. Data will be read and written via the standard Java JDBC API or via the ITA Persistence API if it proves suitable for the Calendar Portlet.



4.5.4 Server Location

Like all portlets, the Calendar Portlet will be deployed on the Portal's WebSphere Application Server.

4.6 Feedback Portlet

4.6.1 Functional Definition

The Feedback Portlet provides a static page defining the process for reporting feedback, via telephone or via email. For email feedback the email functionality is provided by the client browser configuration. There is no server side email capabilities utilized.

4.6.2 External Interfaces

The Feedback Portlet like all portlets provides no unique interface of its own. Because portlets are implemented as Java Servlets they follow the Java Servlet API. The Feedback Portlet can be called from a Template JSP via the include file tag since it is statically rendered:

```
<jsp:include file="/FeedbackPortlet" />
```

4.6.3 Content Residence

There is no content stored in the Feedback Portlet.

4.6.4 Server Location

Like all portlets, the Feedback Portlet will be deployed on the WebSphere Application Server.

4.7 Headlines Portlet

4.7.1 Functional Definition

The Headlines Portlet provides a headline capability. It can retrieve static headline data and display this data on the Schools Portal Home Page or on a logged on users Home Page.

4.7.2 External Interfaces

The Headlines Portlet like all portlets provides no unique interface of its own. Because portlets are implemented as Java Servlets they follow the Java Servlet API. The Headlines Portlet can be called from a Template JSP via the include page tag since it is dynamically rendered:

```
<jsp:include page="/HeadlinesPortlet" />
```



4.7.3 Content Residence

Headlines will be stored in an Oracle Database. Data will be read and written via the standard Java JDBC API or via the ITA Persistence API if it proves suitable for the Headlines Portlet.

4.7.4 Server Location

Like all portlets, the Headlines Portlet will be deployed on the Portal's WebSphere Application Server.

5 RCS (Reusable Common Services) Overview

Please refer to the ITA R2.0 Technical Specification documentation for further details.