DATA-DRIVEN USER INTERFACES USING ORACLE PORTAL

T. Pal
Paul Scherrer Institute, 5232 Villigen, Switzerland

ABSTRACT

Oracle Portal is being used as a development platform for applications requiring relational database access for the control system and operation of the accelerator facilities at the Paul Scherrer Institute (PSI). The underlying technology is based on portlets and dynamic, thin client HTML representation of the database information. Prototype, reusable code constructs written in PL/SQL and JavaScript have been developed.

INTRODUCTION

Web application development for the control and operation of the accelerator infrastructure requires the creation of Graphical User Interfaces (GUIs), with access to real-time variables as well as to data stored in a relational database. This is an important customization effort, involving multiple design, coding, testing and debugging cycles.

Although custom applications are difficult as well as resource intensive to include in a portal, they are the most sought-after. To achieve this, Oracle Portal [1] is used for the development of content-driven websites that reside entirely in an Oracle database. These sites are composed of components called portlets that provide access to web-based resources. Web pages, forms, reports, menus, etc are accessed through a portlet, which can be customized and managed as a component within Oracle Portal.

IMPLEMENTATION

One of the primary objectives is to provide a versatile set of interactive Web applications to access the data and display the query results. Furthermore, a requirement in building the GUI's is interportlet communication – where one portlet triggers an event and updates another portlet on the same page. However, due to the limitations of the wizard building tools of Portal, as well as the inherent limitations of the Web (such as its stateless nature), it is difficult to store and maintain variables. To overcome the aforementioned problems, and incorporate the desired features required by the end-users [2], PL/SQL stored procedures, JavaScript and Oracle's Portal Development Kit (PDK) are exploited. These features have been successfully implemented, albeit preliminary at present, for specific applications [3].

The Portal framework also provides additional services including Single Sign On (SSO), content classification, search and security. Access to the development tools is platform independent.

Portal and Portlets

The browser-based software environment allows developers to build and deploy complex applications, with security features encompassing a framework for delivering access to distributed services and information resources. It employs an entirely web-based environment, and resides as PL/SQL packages in an Oracle database schema. The basics structure and various components available to build a Portal application are illustrated schematically in Figure 1.

The building tools (wizards) allow dynamic HTML representation of the database information. Components such as data entry forms, reports, charts, calendars, menus and hierarchies are implicit in figure 1. The sites are composed of the above mentioned reusable components, and are called portlets. Likewise, tools are provided that allow users to browse database objects (i.e. tables, views, PL/SQL procedures and functions, indexes, sequences, synonyms, etc).

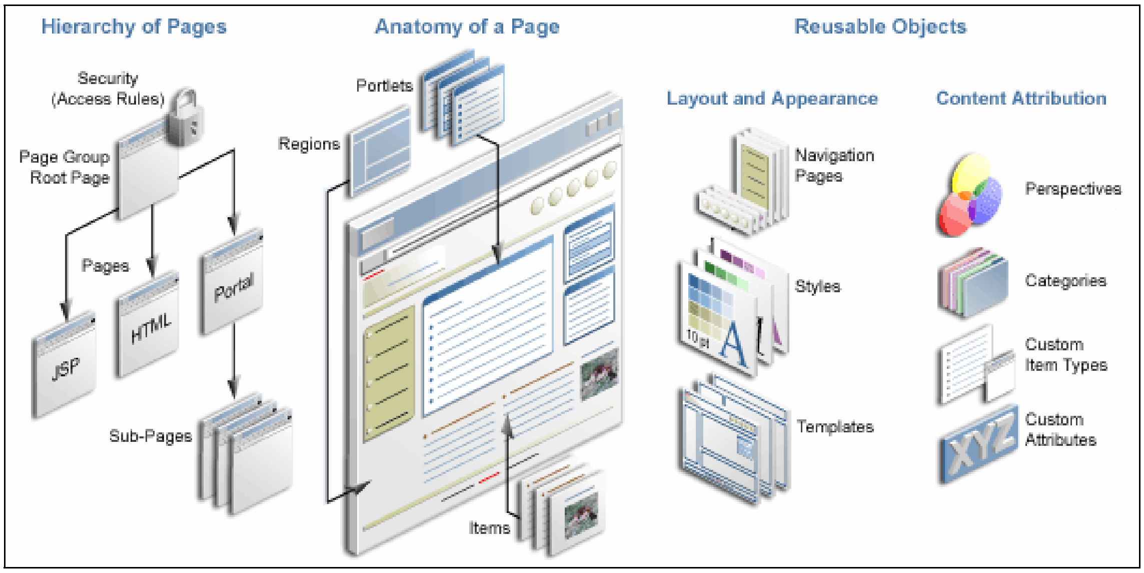


Figure 1: Components of a typical Portal page.

Three-Tier Architecture

Although the product is conceptually referred to as a single entity, it is in fact composed of three distinct logical tiers:

Server-Engine Tier – The Portal part consists of PL/SQL packages and Application Programming Interfaces (APIs), installed in a schema. Database objects store configuration information for each user and the sites in general.

Application-Server Tier – Also referred to as the middle tier. The Oracle Application Server (*9i*AS or *10g*AS) translates client browser requests, which are received as URLs, into database procedural calls. The resulting HTML code which is generated is then passed back to the client browser. The principal component is Oracle's implementation of the Apache listener, labeled as (Oracle) HTTP Server. A PL/SQL Gateway is used for communicating with the Server-Engine.

Client-Browser Tier – This is the presentation and rendering layer for end-user interactions. The interactions can be a straightforward hyperlink click, or a complex multi-field input form for processing, with interactive feedback.

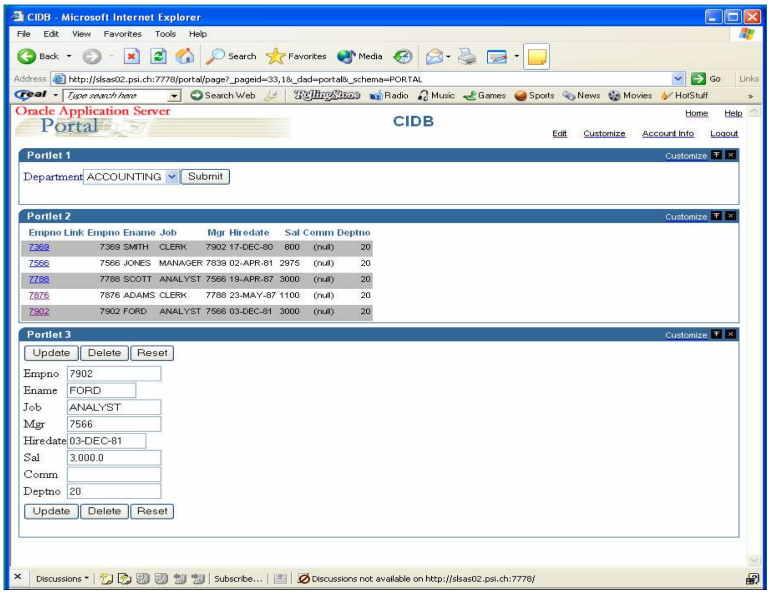
Advanced Functionality

Typical applications under consideration, require between five and eight interactive steps via form and report components to fully process a transaction during inserts or updates. Portal components and portlets offer added value to the end users, such as to significantly reduce the number of navigation steps, to different pages, for completion of a particular task.

The essence of the problem to build these lightweight, yet fully functional applications lies in the inherent stateless nature of the Web. Namely, it is difficult to store variables that can subsequently be referenced, at any point in time. By using the methods of the session storage API in the PDK, allows an application to store variables for the current session in the database, and to reference these variables, once they have been defined for the current session. A specific session store is only available to the current user in the current session [4]. The session values are lost once the user exits the browser or terminates the session.

Generic Prototype

As an illustrative example, a page consisting of three inter-communicating portlets is shown in Figure 2. The upper portlet represents the topmost information of a hierarchical structure of data (generic), for which the drill-down details are displayed and can be modified in the middle and lower components respectively.


CIDB - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://slsas02.psi.ch:7778/portal/page?_pageid=33,1&_dad=portal&_schema=PORTAL
Oracle Application Server Portal
CIDB
Home Help
Edit Customize Account Info Logout
Discussions not available on http://slsas02.psi.ch:7778/

Portlet 1

Department ACCOUNTING Submit

Portlet 2

Empno Link	Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	Deptno
7369	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20
7566	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20
7788	7788	SCOTT	ANALYST	7566	19-APR-87	3000	(null)	20
7876	7876	ADAMS	CLERK	7788	23-MAY-87	1100	(null)	20
7902	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20

Portlet 3

Update Delete Reset

Empno	7902
Ename	FORD
Job	ANALYST
Mgr	7566
Hiredate	03-DEC-81
Sal	3,000.0
Comm	
Deptno	20

Update Delete Reset

Figure 2: Portlet components on a Portal Page.

More specifically, upon change of the value in the combo-box of portlet 1, a JavaScript event handler passes the input parameter(s) to the middle component (portlet 2), which is a report. After a page refresh, the results are displayed on the same HTML page. Similarly, for the lower component (portlet 3), which is a form component, and serves to display the details of a record selected in portlet 2 via a hyperlink. Portlet 3 also contains buttons in order to modify or delete the displayed record.

As the preceding example illustrates, portlet-to-portlet communication is based on certain rules and the parameter values stored. The code snippet below is embedded in the PL/SQL (insert) event handler of the form component for portlet 1:

```
declare
  v_param    number;
  v_session   portal.wwsto_api_session;
begin
  v_param    := p_sesion.get_value_as_number (
                    p_block_name     => ‘DEFAULT’,
                    p_attribute_name => ‘A_VAL_IN’);
  v_session  := portal.wwsto_api_session.load_session(‘LABEL_1’, ‘LABEL_2’);
  v_sesson.set_attribute(‘PARAM’, v_param);
  v_sesion.save_sesion;
end;
```

It serves to illustrate the relative simplicity by which the field values can be extracted dynamically and stored by reference to an alias in a given session, identified by a set of labels. Similarly, to get values, an example is shown in the function below, stored in the database, and is reusable in its invocation:

```
create or replace function function_name return number as
  v_session  portal.wwsto_api_session;
  v_return    number;
begin
  v_session  := portal.wwsto_api_session.load_session(‘LABEL_1’, ‘LABEL_2’);
  v_return    := v_session.get_attribute_as_number(‘PARAM’);
  return(v_return);
end;
```

Noteworthy is also the (Oracle) Portal's implementation and usage of the direct access URL or path aliasing that end-users and developers alike can enter to get to a specific item, document, page, category, or perspective. This is formed as follows:

http://<hostname>:<portnumber>/pls/<dad>/url/page/<pagegroupname>/<objectname>

where all variables have their usual meaning, and <*dad*> is the database descriptor, which contains information on how to connect to the database portal instance.

SUMMARY AND CONCLUSION

Delivering custom applications, within a rapid time scale is a challenge, given the resource typically available. Empirically, the development and deployment of GUIs have taken the majority fraction (approximately two-thirds) of the total time resource. From our experience, it is unlikely that this fraction can be reduced dramatically in the future, as it is dominated by the development and testing time for the site-specific requirements. However, the use of proprietary tools supplied with Oracle Portal, in conjunction with the usage of JavaScript allow us to reconcile the often conflicting requirements of end-user customization with respect to ease of development for applications, requiring database access in addition to real-time variables, in the controls and operations of the accelerator complex at the PSI.

REFERENCES

[1] Oracle Corporation, http://www.oracle.com

[2] "Database Applications for Control System Hardware Management", T. Pal *et al.*, PSI Scientific and Technical Report 2003, Volume VI, Villigen, Switzerland, ISSN 1423-7350, March 2004

[3] "Data-Driven User Interfaces Using Oracle Portal", T. Pal *et al.*, PSI Scientific and Technical Report 2004, Volume VI, Villigen, Switzerland, ISSN 1423-7350, March 2005.

[4] "Oracle *9i* Application Server and Oracle Portal", P. Brassington, MFG Systems Corporation, USA (2002).