Penny's Portal Techniques – Creating Simple Applications with Portal Components

Introduction

There are three main methods of creating applications in portlets:

Custom PL/SQL Portlets
These use the traditional PL/SQL gateway approach to deliver HTML using PL/SQL packages. This approach is convenient for sites which do not have Java skills. I have tended to use this approach for simple portlets that use information from the Portal repository tables, such as the dynamic menu portlet described in the previous edition of Foresight.

(Note: For those of you who were confused by the heading "Creating the Dynamic Men" in the previous article, please note that this was a formatting error and not indicative of any exciting and useful discovery made by the author.)

Custom Java Portlets
These deliver Java servlets or JSPs as portlets. At Sage we have tended to use these for more complex applications requiring a higher level of functionality.

Portal Components
The Oracle Portal product provides a simple wizard style approach for building a number of types of components including Forms, Reports, Calendars, and some really ugly bar graphs. The functionality is limited, but easy and fast to create.

This article describes some of the limitations of the Portal components and some techniques for working around those limitations. It aims to assist you in building simple screens in Oracle Portal without needing to build custom portlets. Since the reports functionality provided is sufficient for most requirements, I focus on the development of data entry screens.

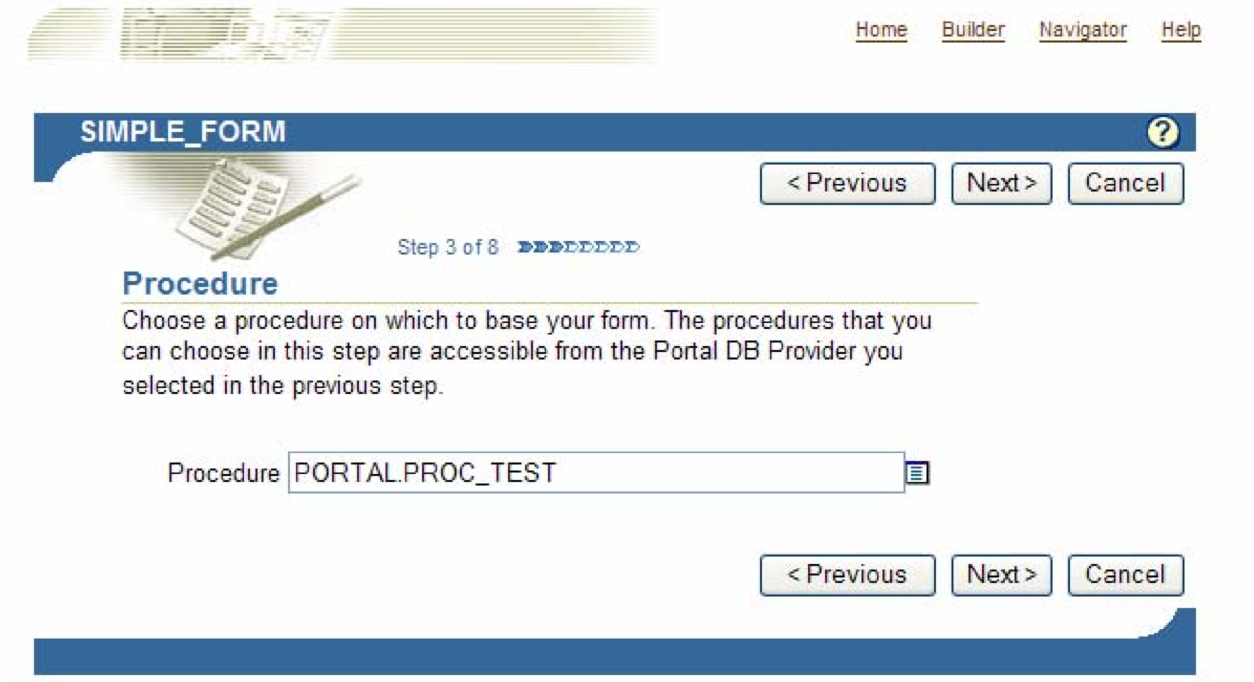
Creating a Simple Form

Types of Forms
Simple Forms are created in the Providers tab from the Locally Built Provider link. A database provider is created which serves as the containing application. Three types of form can be created:

Form based on a table or view
For those of you who are Oracle Forms Developers, think of this as a simple database block.

Master Detail form
This provides a form with a single record master, and multi record detail. I find it really quite ugly and almost unusable.

Form based on a procedure
This is similar to an Oracle Forms Developer block based on a procedure. It provides a more flexible approach for simple data entry forms and is the approach I would recommend. The screen just becomes a simple interface for accepting parameters that are then submitted to a PL/SQL procedure. When the Submit button is pressed the packaged procedure is executed. Business rules can be implemented in the procedure.

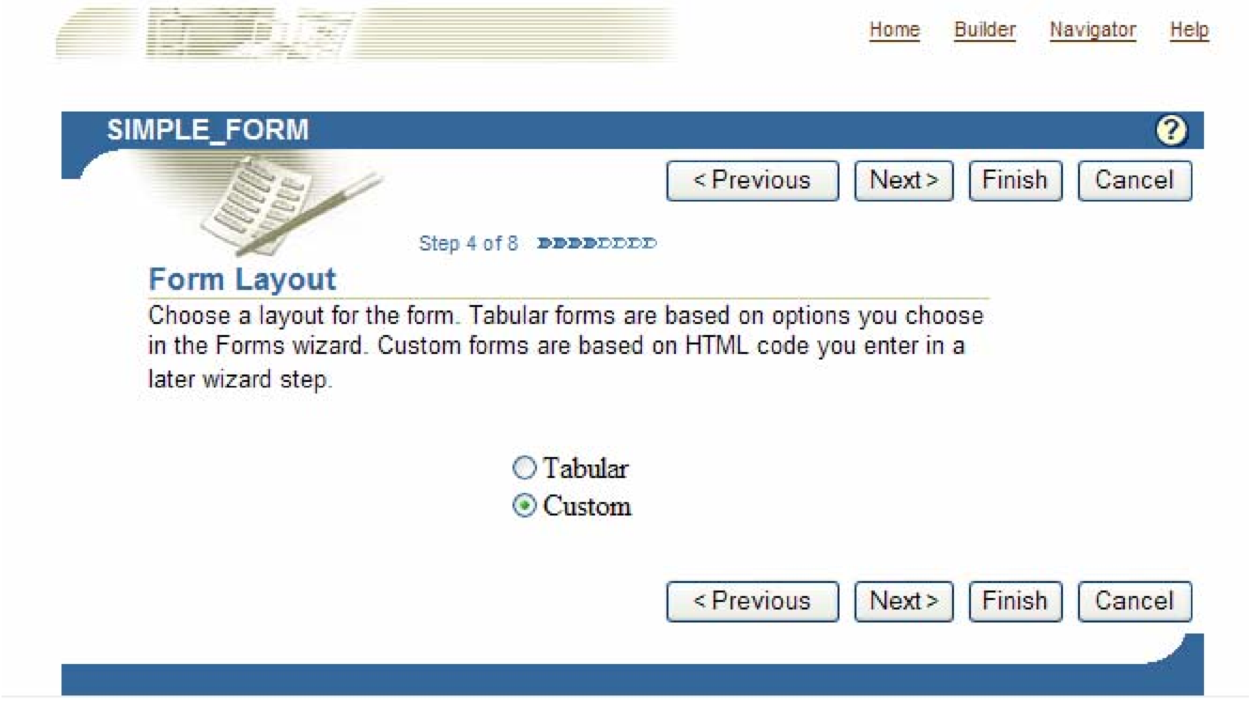


Procedure

Choose a procedure on which to base your form. The procedures that you can choose in this step are accessible from the Portal DB Provider you selected in the previous step.

Procedure: PORTAL.PROC_TEST

Create the form layout as a Custom Layout since this allows us to manually amend the HTML code to create the layout we require.

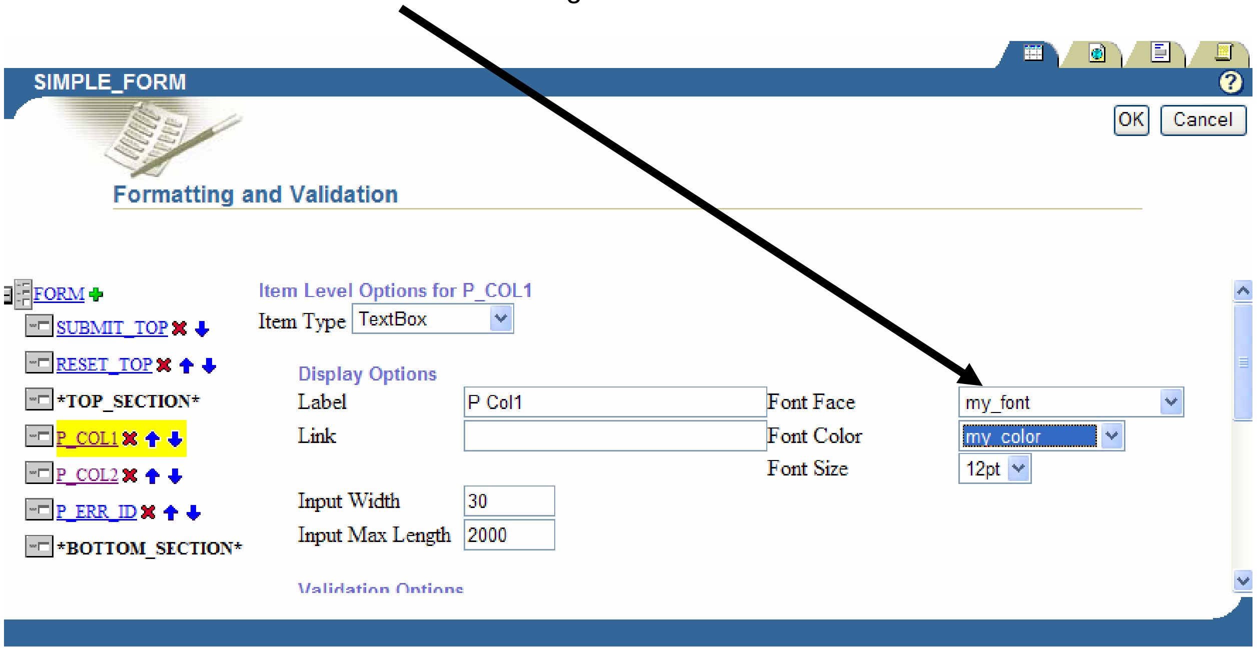


Form Layout

Choose a layout for the form. Tabular forms are based on options you choose in the Forms wizard. Custom forms are based on HTML code you enter in a later wizard step.

Tabular [✓ Custom]

I would recommend that you create a custom font and colour (under the Shared Components section from the Locally Built Provider link of the Provider tab) rather than choosing a standard colour or font. This makes it easier to change the colour or font later on.



Item Level Options for P_COL1

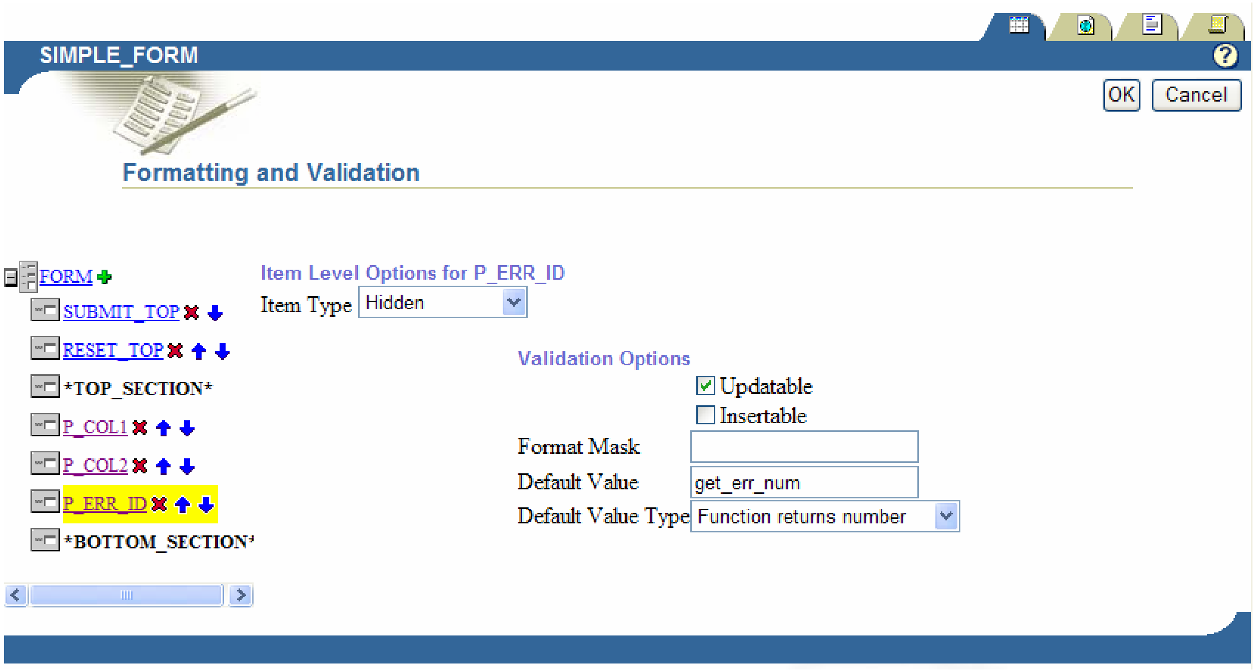
Item Type: TextBox

Display Options

Label	P Col1	Font Face	my_font
Link		Font Color	my_color
		Font Size	12pt
Input Width	30		
Input Max Length	2000		

Validation Options

Error Handling
An additional parameter is added to the procedure. This contains a unique error number generated from a sequence, supplied as a default value in the form.



Item Level Options for P_ERR_ID

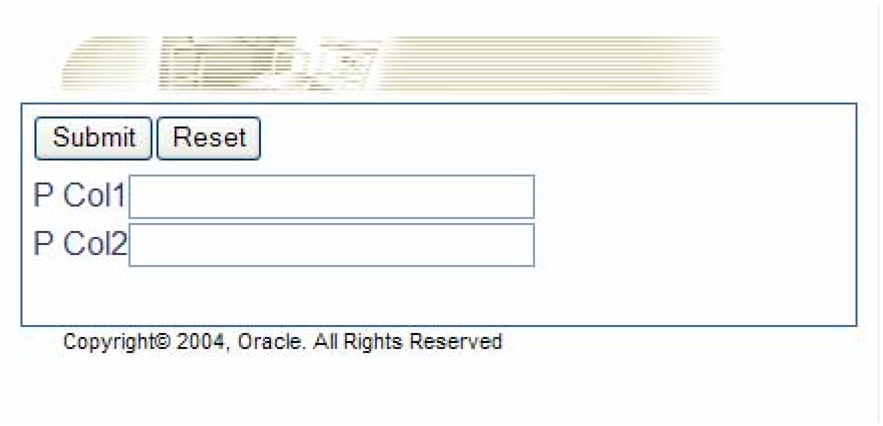
Item Type: Hidden

Validation Options

☑ Updatable
☐ Insertable

Format Mask	
Default Value	get_err_num
Default Value Type	Function returns number

The Simple Screen



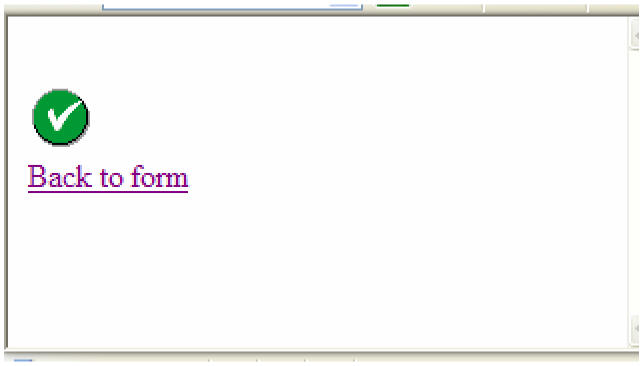
P Col1

P Col2

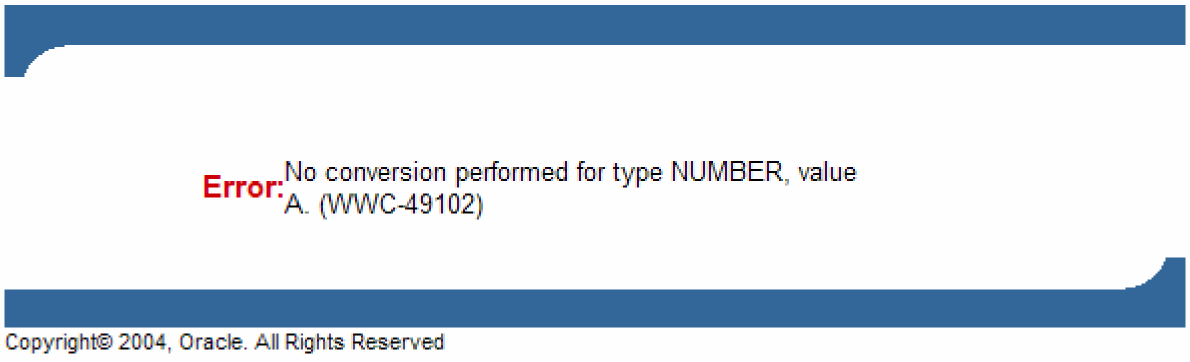
Copyright© 2004, Oracle. All Rights Reserved

What happens after the screen is successfully submitted?
Once the screen is submitted the default behaviour is a problem. If the screen is successful we have a screen with a large tick displayed. This is not either attractive or acceptable to most sites. If the procedure raised an exception an ugly error message is displayed. The technique described below is one we have used successfully to work around this problem for a simple feedback form.

Successful Submission



Unsuccessful Submission

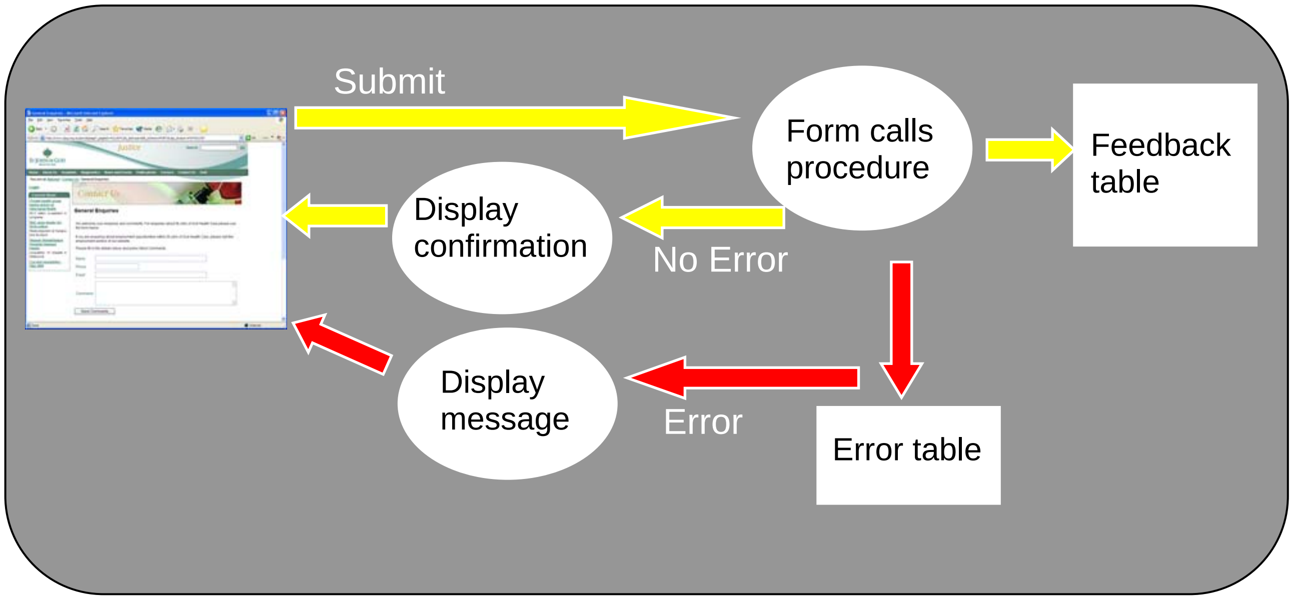


Error: No conversion performed for type NUMBER, value A. (WWC-49102)

Copyright© 2004, Oracle. All Rights Reserved

Handling Success and Failure

The following technique is used to handle successful and failed submission in a more user friendly manner.



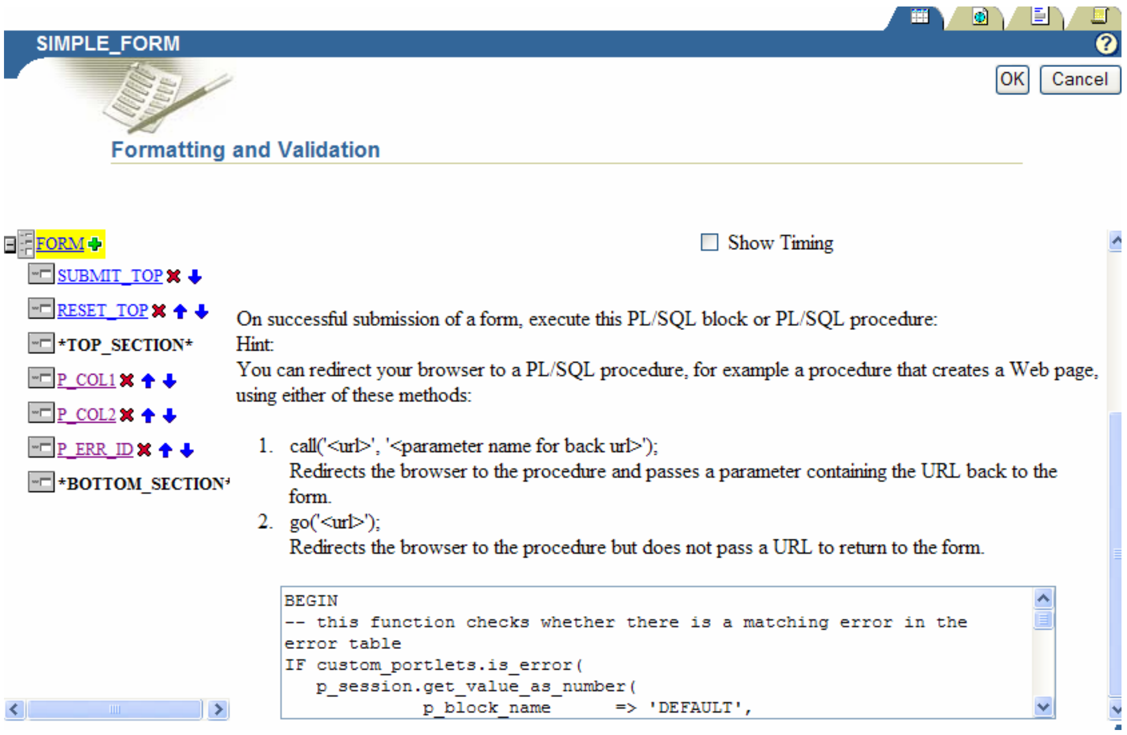
The procedure called by the Portal form should have an exception handler which inserts an error into the error table using the ERR_ID as a key.

After the form is submitted we check whether there is an error in the table which matches the forms error number. This determines the value of a parameter passed to a dynamic PL/SQL portlet.

Listing 1: Call this code after the form has been executed

```
BEGIN
-- this function checks whether there is a matching error in the error table
IF custom_portlets.is_error(
   p_session.get_value_as_number(
             p_block_name      => 'DEFAULT',
             p_attribute_name  => 'A_P_ERR_ID'))
THEN
   -- If there is an error we pass ERROR in the p_result parameter
   go('PORTAL_ADMIN.DYN_DISPLAY_CONFIRM.show?
   p_arg_names=_title&p_arg_values=YES&p_arg_names=p_result&p_arg_values=ERROR
   &p_arg_names=p_current_location&p_arg_values=URL/PAGE/PAGEGROUPNAME/
   PAGENAME');
ELSE
   --Otherwise we pass SUCCESS in the p_result parameter
   go('PORTAL_ADMIN.DYN_DISPLAY_CONFIRM.show?
   p_arg_names=_title&p_arg_values=YES&p_arg_names=p_result
   &p_arg_values=SUCCESS&p_arg_names=p_current_location&p_arg_values=
   URL/PAGE/PAGEGROUPNAME/PAGENAME');
END IF;
END;
```

The code in Listing 1 is entered in the Edit Form screen as shown below.



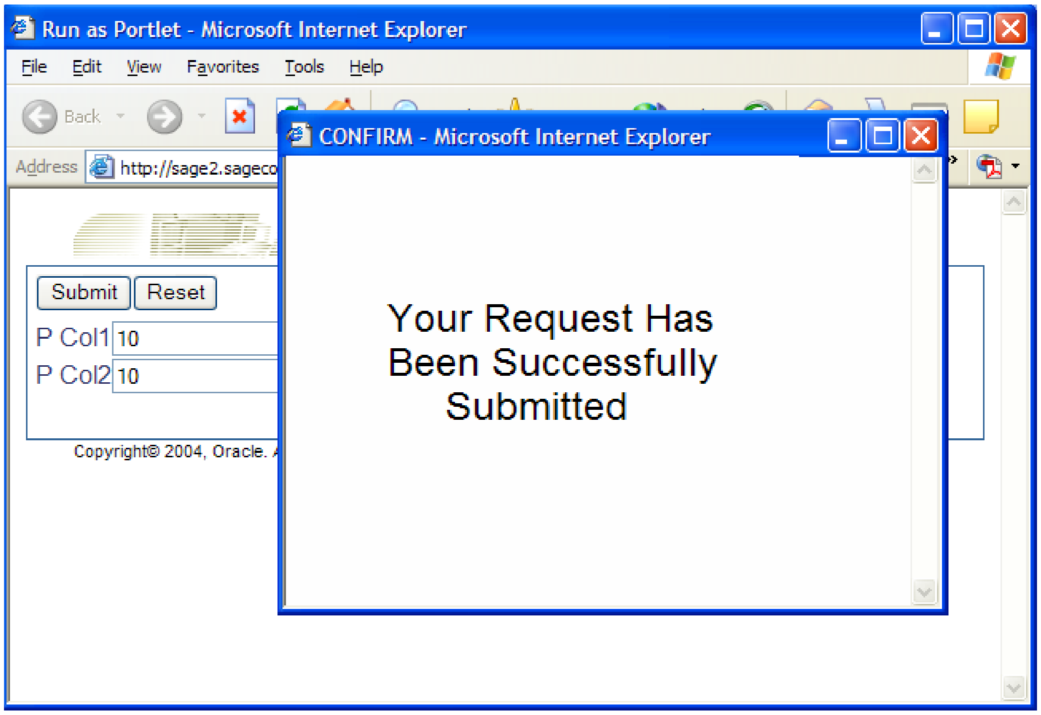
On successful submission of a form, execute this PL/SQL block or PL/SQL procedure:
Hint:
You can redirect your browser to a PL/SQL procedure, for example a procedure that creates a Web page, using either of these methods:

1. call('<url>', '<parameter name for back url>');
   Redirects the browser to the procedure and passes a parameter containing the URL back to the form.
2. go('<url>');
   Redirects the browser to the procedure but does not pass a URL to return to the form.

```
BEGIN
-- this function checks whether there is a matching error in the
error table
IF custom_portlets.is_error(
   p_session.get_value_as_number(
             p_block_name      => 'DEFAULT',
```

The dynamic portlet calls a database procedure. This displays the original page and then takes appropriate action depending on whether an error condition has been indicated.

Customised Successful Submission



Your Request Has Been Successfully Submitted

Listing 2: Dynamic portlet to call the redirect_confirm database procedure

```
-- Dynamic page - redirect confirm
<ORACLE>
BEGIN
   custom_portlets.redirect_confirm(:p_current_location, :p_result,);
END;
</ORACLE>
```

If an error exists a custom error screen is displayed. This can be created as a simple report component. When the user acknowledges the message the original screen is redisplayed.

If there is no error record in the error table a success screen is displayed. This is the same report component, but which displays an error message if the p_result parameter = ERROR. When the user acknowledges the success or error message control is returned to the original screen.

Listing 3: Display the original page and then pop up a success or error report

```
-- Database procedure to display the confirm or error screen
PROCEDURE redirect_confirm (p_current_location IN VARCHAR2, p_result IN
VARCHAR2)
IS
-- RPT_CONFIRM is a simple report component that expects a p_result parameter
-- that determines the message that will be displayed
-- The report is displayed in a separate popup window.
BEGIN
   htp.p('
   <html>
   <head>

   <script type="text/javascript">
   function goconfirm()
   {
   location="'||p_current_location||'";
   window.open("PORTAL_ADMIN.RPT_CONFIRM.show?p_arg_names=’||
   ‘_show_header&p_arg_values=YES'||
   '&p_arg_names=_max_rows&p_arg_values=20&p_arg_names=_portal_max_rows&'||
   'p_arg_values=20&p_arg_names=p_type&p_arg_values='||p_result||
   '","Confirm","toolbar=no, location=no, top=300, left=300,'||
   ' directories=no, status=no, menubar=no, scrollbars=no, resizable=no, '||
   'copyhistory=yes, width=300, height=200")
   }
   </script>
   </head>
   <body onload="goconfirm()">
   </body>
   </html>');

END redirect_confirm;
```