Querries with POWL (Personal Object Work List) and Web Dynpro ABAP

Applies to:
Web Dynpro ABAP, ABAP Developers. For more information, visit the Web Dynpro ABAP homepage.

Summary
As we know, in Web Dynpro ABAP we can use the Select Options and the ALV component to offer to the end user advanced capability of selection and capacity to display the search result into an ALV table. With the help of the POWL, we can offer select options for the end user, we can offer advanced configuration possibilities, we can use the authorization object CA_POWL to restrict the users’ access, the search result will be shown into an ALV table. Main advantages are the reduction of the programming effort and many other additional capabilities centralised into a so-called feeder class.

The scope of this article is to present the way we can use queries with POWL list.

For this purpose, we are going to create a step-by-step application.

Authors: Dr. Cristea Ana Daniela, Memmer Manfred, Thilo Klopfer
Company: Cellent AG
Created on: 28th April 2011

Author Bio
Dr. Cristea Ana Daniela has worked as an Assistant Professor at the Polytechnic University of Timișoara (Romania) for 4 years. The subject of her doctoral dissertation was “Contribution to creating and developing a new SAP authorization concept based on RFID”. She is the author and coauthor of 4 books (Web Dynpro ABAP for Practitioners, Computer Programming, Computer Utilization, Interface and Peripheral) and many articles. Currently, she is working as a SAP consultant. She was named in the “Who’s Who in Science and Engineering 2011 / 2012”.

Memmer Manfred is a SAP Consultant and has more than 18 years of IT experience, of which more than 11 years with SAP. He has knowledge in APAP, Web Dynpro ABAP and BSP programming, XML interfaces, creating Adobe forms with the Adobe Live Cycle Designer, developing of functions in the BRF plus (Business Rule Framework plus), developing and customizing of SAP SRM 5.0-7.0 for customers, and so on.
Table of Contents

Requirements ........................................................................................................................................ 3
Creating the feeder class .................................................................................................................... 4
  Method IF_POWL_FEEDER~GET_SEL_CRITERIA ........................................................................ 5
  Method IF_POWL_FEEDER~GET_OBJECTS .................................................................................. 8
  Method IF_POWL_FEEDER~GET_OBJECT_DEFINITION ............................................................ 9
The transactions we need to connect the feeder class with the Web Dynpro application .................. 10
Testing the POWL ............................................................................................................................. 13
  Testing with the help of our Web Dynpro application ............................................................... 13
  Using POWL Web Dynpro application ....................................................................................... 13
Conclusions .......................................................................................................................................... 16
Related Content .................................................................................................................................. 17
Disclaimer and Liability Notice .......................................................................................................... 18
Requirements

For purposes of the present article we want to create an application to offer capabilities to select three fields of a table, to offer possibilities to customise the selection mode (including possibility to provide certain values in fields) and to display the result in an ALV table. For this, we use POWL. What we get to the end of the application is presented in Fig. 1.

Fig. 1 The result that we will get

Data selection will be made in the database table YLIBRARY, presented in Fig. 2.

Fig. 2 Database table used for our example
The Book_id field has its own defined Search help (Fig. 3). We will use this search help for the selection criteria of the POWL.

![Search help](image)

**Creating the feeder class**

A feeder class is a normal ABAP class that implements the interface IF_POWL_FEEDER. We can find more details about this class and its methods at the reference [1]. By using the methods of this class, we will be able to:

- interact with the backend;
- define the fields which will be available into the Select Options;
- define the ALV settings, etc.

Our feeder class YPOWL is presented in Fig. 4.
Fig. 4 Our feeder class YPOWL has as interface IF_POWL_FEEDER

All we have to do is to implement its methods depending on application requirements.

**Method IF_POWL_FEEDER~GET_SEL_CRITERIA**

We need this method to define the selection criteria which we will have into the Web Dynpro application, and to define in the same time the selection criteria that can be used later to define new queries.

Into this method, we implement the code presented in Listing 1.

**METHOD if_powl_feeder~get_sel_criteria.**

DATA: ls_selcrit TYPE powl_selcrit_sty,
     ls_default TYPE rsparams.

CLEAR c_selcrit_defs.
CLEAR c_default_values.
ls_selcrit-selname = 'BOOK_ID'.
ls_selcrit-kind = 'S'.
ls_selcrit-param_type = 'I'.
ls_selcrit-selopt_type = 'A'.
ls_selcrit-allow_admin_change = 'X'.
ls_selcrit-datatype = 'YID_BOOK'.
ls_selcrit-ddic_shlp = 'YSH_YLIBRARY'.
INSERT ls_selcrit INTO TABLE c_selcrit_defs.
CLEAR ls_selcrit.
ls_selcrit-selname = 'AUTHOR'.
ls_selcrit-kind = 'S'.
ls_selcrit-param_type = 'I'.
ls_selcrit-selopt_type = 'A'.
ls_selcrit-allow_admin_change = 'X'.
ls_selcrit-datatype = 'YAUTHOR'.

<table>
<thead>
<tr>
<th>Class Interface</th>
<th>YPOWL</th>
<th>Implemented/Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interfaces</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aliases</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Level</th>
<th>Mvl.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF_POWL_FEEDER~GET_ACTIONS</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Auftragsmetadaten für den dargestellten Objektyp definieren</td>
</tr>
<tr>
<td>IF_POWL_FEEDER~GET_ACTION_CONF</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Aktionenbestätigungsmeldung einer Aktion</td>
</tr>
<tr>
<td>IF_POWL_FEEDER~GET_SEL_CRITERIA</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Metadaten der Selektionskriterien definieren</td>
</tr>
<tr>
<td>IF_POWL_FEEDER~GET_FIELD_CATALOG</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Metadaten des Feldkatalogs definieren</td>
</tr>
<tr>
<td>IF_POWL_FEEDER~GET_OBJECT_DEFINITION</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Datenstruktur für dargestellten Objektyp definieren</td>
</tr>
<tr>
<td>IF_POWL_FEEDER~GET_OBJECTS</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Datenbeschaffung für dargestellten Objektyp</td>
</tr>
<tr>
<td>IF_POWL_FEEDER~GET_DETAIL_COMP</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Objekt-detaillierte WD-Komp., die IF_POWL_DETAIL implementiert</td>
</tr>
<tr>
<td>IF_POWL_FEEDER~HANDLE_ACTION</td>
<td>Insta.</td>
<td>Pub.</td>
<td>Aktionen für Objektyp (vgl. GET_ACTIONS) bearbeiten</td>
</tr>
</tbody>
</table>
Listing 1 Method if_powl_feeder~get_sel_criteria

As we have seen, by using the SAP structure POWL_SELCRIT_STY (Fig. 5) we can customize each field that we offer as selection criteria on the screen (DATA ls_selcrit TYPE powl_selcrit_sty).

Fig. 5 Structure POWL_SELCRIT_STY
As selection criteria, we have: Parameter or Select option (Fig. 6). In our case, we have used the selection criteria S.

<table>
<thead>
<tr>
<th>Domain</th>
<th>POWL_SELKIND_DOM</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Description</td>
<td>kind of selection criteria: Parameter or Select option</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6 Selection criteria

The parameter type can be:
I – Input field
C – Checkbox
D – Dropdown list
T – Textline, (See Domain Fig. 7).

In our case, we have used the parameter type input fields.

<table>
<thead>
<tr>
<th>Domain</th>
<th>POWL_CRITYPE_PARAM_DOM</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Description</td>
<td>display type for simple parameters</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 7 Parameter type

Select Options display style:
A – Select-option with interval and multi-selection
I – Select-option without multi-selection
M – Select-option without interval (Defined in domain, Fig. 8).
For the Book_id field, we have used the defined search help: ls_selcrit-ddic_shlp = 'YSH_YLIBRARY'.

**Method IF_POWL_FEEDER~GET_OBJECTS**

We have to implement this method to be able to select data from our database table YLIBRARY, based on search criteria. This method will be called every time the end user presses the Search button or the Refresh button. Before we implement its coding, we create an attribute (Fig. 9) required to hold the data read from the database table YLIBRARY and to define the data type definition of the actual query (used later in the method: if_powl_feeder~get_object_definition).

In this method, we implement the code presented in Listing 2.

**METHOD if_powl_feeder~get_objects.**

```
DATA: lt_book_id TYPE RANGE OF yid_book,
     lt_author TYPE RANGE OF yauthor,
     lt_price TYPE RANGE OF yprice,
     ls_book_id LIKE LINE OF lt_book_id,
```
Is_author LIKE LINE OF lt_author,
Is_price LIKE LINE OF lt_price.
FIELD-SYMBLOS: <lfs_params> TYPE rparams,
               <lf_result> TYPE ylibrary.
" Fill the range tables
LOOP AT i_selcrit_values ASSIGNING <lfs_params>.
CASE <lfs_params>-selname.
   WHEN 'BOOK_ID'.
     MOVE-CORRESPONDING <lfs_params> TO ls_book_id.
     INSERT ls_book_id INTO TABLE lt_book_id.
   WHEN 'AUTHOR'.
     MOVE-CORRESPONDING <lfs_params> TO ls_author.
     INSERT ls_author INTO TABLE lt_author.
   WHEN 'PRICE'.
     MOVE-CORRESPONDING <lfs_params> TO ls_price.
     INSERT ls_price INTO TABLE lt_price.
ENDCASE.
ENDLOOP.
REFRESH me->mt_result.
SELECT * FROM ylibrary INTO TABLE me->mt_result
   WHERE book_id IN lt_book_id
       AND author IN lt_author
       AND price IN lt_price.
e_results = me->mt_result.
ENDMETHOD.

Listing 2 Method if_powl_feeder~get_objects

P.S. In this simple example, we used CASE to take over for each field the range table containing the search information inserted by the user. For this purpose, we can create a dynamic variant, for avoiding the situation of having a CASE with too many conditions.

Method IF_POWL_FEEDER~GET_OBJECT_DEFINITION
The implementation of this method is presented in Listing 3.
method if_powl_feeder~get_object_definition.
   e_object_def = cl_abap_tabledescr->describe_by_data( me->mt_result ).
endmethod.

Listing 3 if_powl_feeder~get_object_definition
The transactions we need to connect the feeder class with the Web Dynpro application

The transactions required for defining the application id, the type and the queries are presented below. These transactions are:

FPB_MAINTAIN_HIER required for creating an application id (Fig. 10)

POWL_TYPE required for defining a type (Fig. 11)
POWL_TYPER required for connecting the application with the type (Fig. 12)

**Change View "View: Type - Role assignment": Overview**

Fig. 12 Connecting the application with the type

POWL_QUERY required for defining one or more queries. In our case, we define 2 queries (Fig. 13).

Fig. 13 Creating queries

After this, we can create the query parameters and the query settings with the help of the corresponding buttons:

For the query ZLIBRARY_QUERY2, we defined the settings presented in Fig. 14
Fig. 14 Defining selection criteria settings and parameters

At the end, we need only the transaction **POWL_QUERYR** to connect the two created queries with the application (Fig. 15):

**Fig. 15 Connecting the application with the queries**
Testing the POWL

To test the newly created development objects, we can:
- integrate them in our Web Dynpro application, or
- use the SAP Web Dynpro application POWL (Personal Object Worklist).

Testing with the help of our Web Dynpro application

In case we want to create our own Web Dynpro [2-3] application in which to use the created development objects, we have to use the SAP Component POWL_UI_COMPONENT, Fig. 16.

<table>
<thead>
<tr>
<th>Used Components</th>
<th>Implemented interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Used Web Dynpro Components</strong></td>
<td></td>
</tr>
<tr>
<td>Component Use</td>
<td>Component</td>
</tr>
<tr>
<td>POWL</td>
<td>POWL_UI_COMP</td>
</tr>
</tbody>
</table>

Fig. 16 Using the Component POWL_UI_COMP

Then, with the help of a ViewContainerUIElement, we can insert the view POWL_MASTER into our View. At the reference [4], we will find an article that shows step-by-step how we can use this SAP component into our own Web Dynpro application.

Using POWL Web Dynpro application

To use the POWL Web Dynpro application, we go to SAP Easy Access and insert a new Web Dynpro Application into the Favorites folder (Fig. 17).

SAP Easy Access

Fig. 17 SAP Easy Access

After this, we use the Web Dynpro POWL with the parameter applied = our own application id name YLIBRARY_APP, Fig. 18. For configurations, it is available the well known WDCONFIGURATION parameter.
Querries with POWL (Personal Object Work List) and Web Dynpro ABAP

Fig. 18 The Web Dynpro application POWL

The initial screen is presented in Fig. 19.

Fig. 19 – POWL testing

As we can see, in the start screen we can create a new query ("Define new query" link) or we can use the already created queries (ZLIBRARY_QUERY and ZLIBRARY_QUERY2) with the help of the "Personalize" link.

At the beginning, we create a new query named Query1 (Fig. 20):
Fig. 20 Creating the own query with the help of “Define new query” link
With the help of “Personalization” link, we use our own defined queries (Fig. 21).

**Fig. 21 How to use our own query ZLIBRARY_QUERY2**

In this case, only the AUTHOR is visible into the selection screen and has the default values that we have provided for ZLIBRARY_QUERY2 by pressing “Query Parameters” and “Query Settings”.

**Conclusions**

1. Using the feeder class (implementing the IF_POWL_FEEDER interface), we can define how the resulting application will look like. We can define the selection criteria, define the actions, introduce new buttons on the ALV toolbar, define the number of rows and columns, etc., all these by implementing the methods of one single class: the feeder class;

2. With some transactions, we can connect the feeder class with the Web Dynpro application;

3. With the SAP component POWL_UI_COMPONENT, we can integrate the POWL lists in our own Web Dynpro applications;

4. To give authorization for the POWL, we can use the authority object CA_POWL. We can assign this authorization object to a role for creating the authorization of a user (e.g. what application id can use the user, if the user can personalize the query, if the user can create new queries, etc.);

5. We have support for Portal integration.
Related Content

Feeder Implementation


Integrating POWL with Web Dynpro ABAP, Saurav Mago

For more information, visit the Web Dynpro ABAP homepage
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.