Execute a Transaction in the ERP Backend System from a Web Dynpro for ABAP Application via Webgui

Applies to:

SAP Net Weaver 7.0, ABAP. For more information, visit the ABAP homepage.

Summary

This sample code will demonstrate how to execute a transaction in a backend ERP system from a Web Dynpro for ABAP application. This example will display a list of invoices then allow the user to click on an invoice number and display the details of that invoice from the backend ERP system in a web browser.

Author: Richard Koziol

Company: SAP Americas Inc.

Created on: 07 October 2009

Author Bio

Richard Koziol is a technical consultant working in SAP America's consulting division for the past 11 years. Currently based out of the Edison NJ office Richard specializes in all areas of ABAP development including Web Dynpro for ABAP, BSP,classic ABAP as well as SOA and integration technologies.

Table of Contents

Summary

Calling a transaction from an ERP backend system can be easily accomplished thanks to the WebGUI. This can help enhance the user experience by giving visibility to the ERP backend system with a single click and without forcing the user to launch the SAPgui. Here is a concise, all in one place to find a simple approach for this solution.

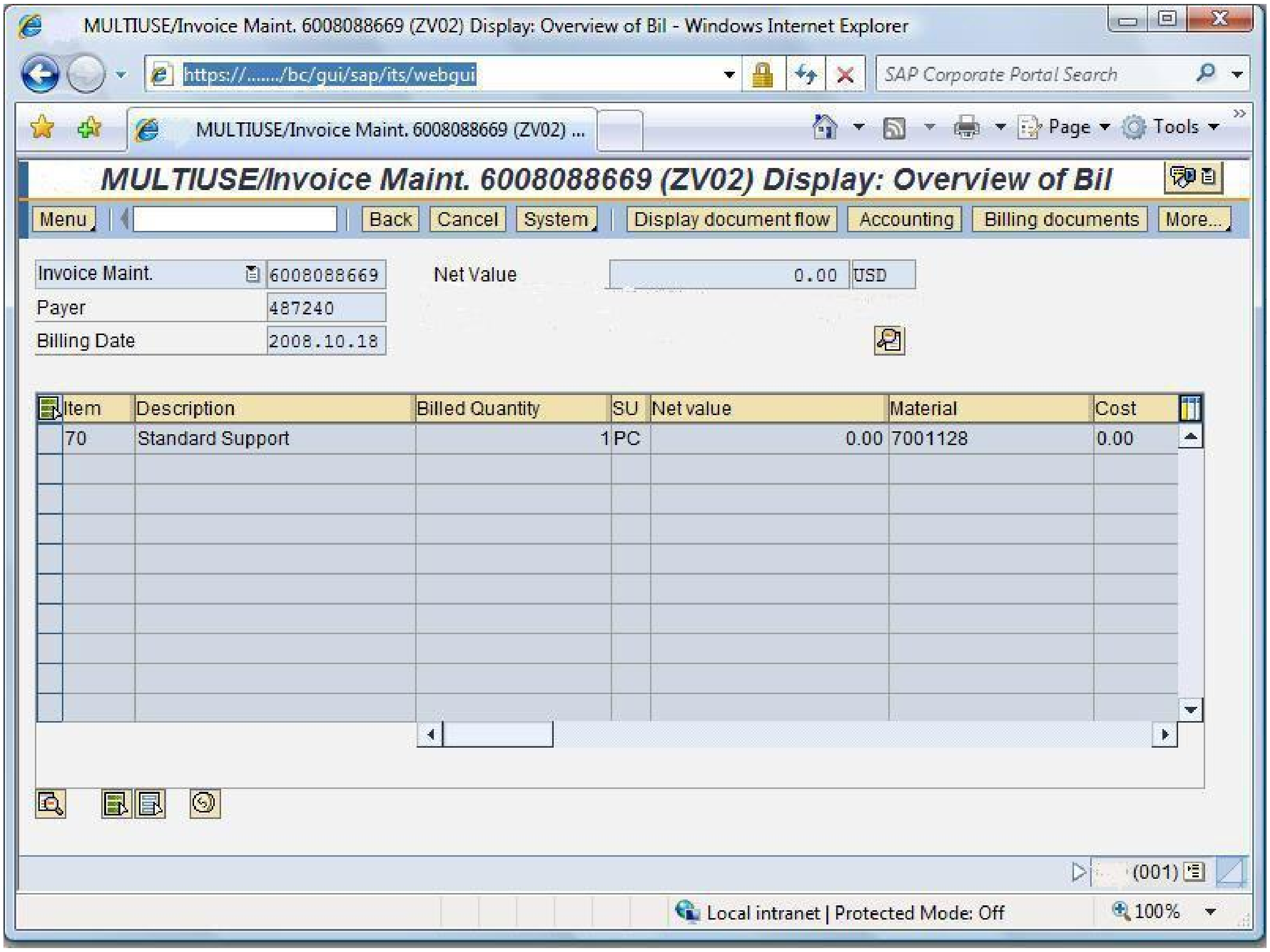
- This document will make use of the ALV. It is assumed that the developer already has a basic understand of using the ALV for Web Dynpro for ABAP

Pre-requisites

- Web Application Server 7.0 or higher
- End users should already have a userid in the backend system
- End users should already have the roles/profiles needed to execute the transaction in the backend system

Overview and Result

Below is a sample of an ERP transaction rendered through the WebGUI after it has been called from a Web Dynpro for ABAP application


MULTIUSE/Invoice Maint. 6008088669 (ZV02) Display: Overview of Bil - Windows Internet Explorer
https://......./bc/gui/sap/its/webgui
SAP Corporate Portal Search
MULTIUSE/Invoice Maint. 6008088669 (ZV02) ...
Page
Tools
Menu
Back
Cancel
System
Display document flow
Accounting
Billing documents
More...
(001)
Local intranet | Protected Mode: Off
100%

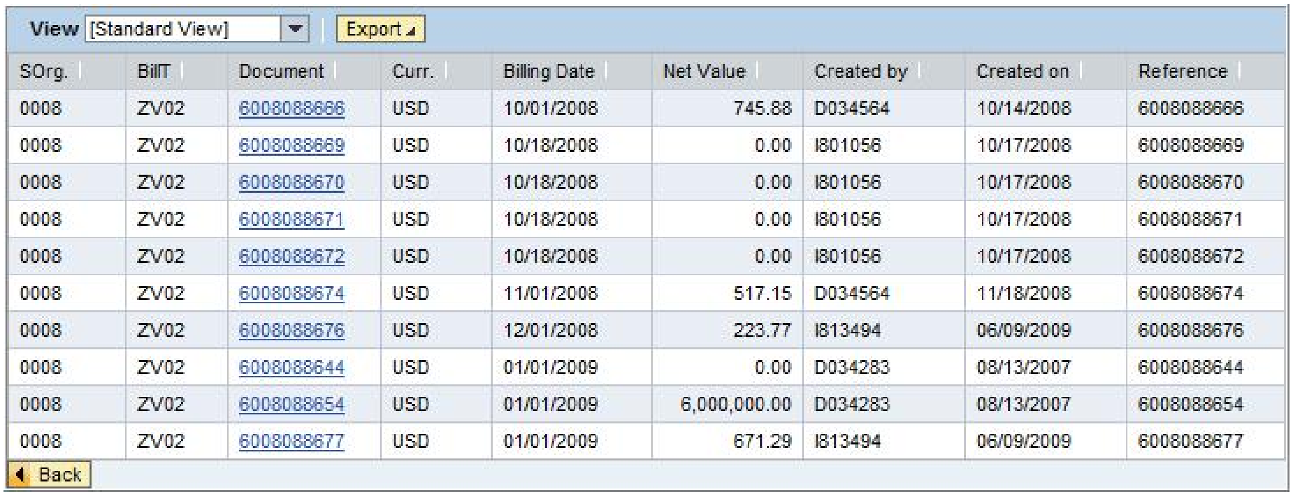
MULTIUSE/Invoice Maint. 6008088669 (ZV02) Display: Overview of Bil

Invoice Maint.	6008088669	Net Value	0.00 USD
Payer	487240		
Billing Date	2008.10.18		

Item	Description	Billed Quantity	SU	Net value	Material	Cost
70	Standard Support	1	PC	0.00	7001128	0.00

Note: Users will have the same authorizations as if they were logged into a backend SAP ERP system via the SAPGUI. The command box is displayed by default. This can be removed via an ITS service parameter ~noheaderokcode = 1

Solution



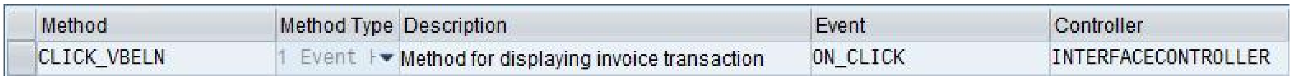
SOrg.	BillT	Document	Curr.	Billing Date	Net Value	Created by	Created on	Reference
0008	ZV02	6008088666	USD	10/01/2008	745.88	D034564	10/14/2008	6008088666
0008	ZV02	6008088669	USD	10/18/2008	0.00	I801056	10/17/2008	6008088669
0008	ZV02	6008088670	USD	10/18/2008	0.00	I801056	10/17/2008	6008088670
0008	ZV02	6008088671	USD	10/18/2008	0.00	I801056	10/17/2008	6008088671
0008	ZV02	6008088672	USD	10/18/2008	0.00	I801056	10/17/2008	6008088672
0008	ZV02	6008088674	USD	11/01/2008	517.15	D034564	11/18/2008	6008088674
0008	ZV02	6008088676	USD	12/01/2008	223.77	I813494	06/09/2009	6008088676
0008	ZV02	6008088644	USD	01/01/2009	0.00	D034283	08/13/2007	6008088644
0008	ZV02	6008088654	USD	01/01/2009	6,000,000.00	D034283	08/13/2007	6008088654
0008	ZV02	6008088677	USD	01/01/2009	671.29	I813494	06/09/2009	6008088677

Here is the code that was used to accomplish this task.

As you can see from the screenshot the output was displayed using the ALV. The WDDOINIT method is executed only once before the view is rendered so the invoice number column is set as a link within this method.

```
METHOD wddoinit .
  DATA: lr_cmp_usage              TYPE REF TO if_wd_component_usage,
        lr_interfacecontroller    TYPE REF TO iwci_wdr_select_options,
        lo_interfacecontroller    TYPE REF TO iwci_salv_wd_table,
        lo_cmp_usage              TYPE REF TO if_wd_component_usage.

  DATA: lr_column                 TYPE REF TO cl_salv_wd_column,
        lr_column_settings        TYPE REF TO if_salv_wd_column_settings,
        lr_table_settings         TYPE REF TO if_salv_wd_table_settings,
        l_value                   TYPE REF TO cl_salv_wd_config_table,
        lt_columns                TYPE salv_wd_t_column_ref,
        ls_column                 TYPE salv_wd_s_column_ref,
        lr_col_header             TYPE REF TO cl_salv_wd_column_header,
        lr_cell_editor            TYPE REF TO cl_salv_wd_uie,
        lr_link                   TYPE REF TO cl_salv_wd_uie_link_to_action,
        lr_text_view              TYPE REF TO cl_salv_wd_uie_text_view,
        lv_columnname             TYPE string,
        lv_otr                    TYPE string.

  lo_cmp_usage = wd_this->wd_cpuse_salv_wd_table_vbrk( ).
  IF lo_cmp_usage->has_active_component( ) IS INITIAL.
    lo_cmp_usage->create_component( ).
  ENDIF.

  lo_interfacecontroller = wd_this->wd_cpifc_salv_wd_table_vbrk( ).
  l_value = lo_interfacecontroller->get_model( ).
  lr_column_settings ?= l_value.
  lr_table_settings ?= l_value.
  lt_columns = lr_column_settings->get_columns( ).

  LOOP AT lt_columns INTO ls_column.

    CASE ls_column-id.

      WHEN 'VBELN'.
        lr_column = lr_column_settings->get_column( 'VBELN' ).
        lr_column->set_width( '10' ).

* Once we encounter the invoice number column set it as a link
        CREATE OBJECT lr_link.
        lr_link->set_text_fieldname( ls_column-id ).
        lr_column->set_cell_editor( lr_link ).
    ENDCASE.
  ENDLOOP.

ENDMETHOD.
```

Now that we have created the invoice number as a link we need to create a method for the ON_CLICK event of the ALV. In this example a method named CLICK_VBELN will be defined as method type 1 Event Handler. Similar to the screenshot below:



Method	Method Type	Description	Event	Controller
CLICK_VBELN	1 Event H	Method for displaying invoice transaction	ON_CLICK	INTERFACECONTROLLER

In order to display the backend transaction we must gather and construct the URL to that transaction and in our example we will display the transaction via the WebGUI.

The flow is as follows:

1. Get the host, protocol and port number
2. Construct the URL to the transaction
3. Call a method to create an external window passing it the newly constructed URL

The code within the method:

```
METHOD click_vbeln.

  DATA: lr_vbeln                     TYPE REF TO vbeln,
        lv_vbeln                     TYPE vbeln,
        lv_url                       TYPE string,
        lv_trans_code                TYPE url_param,
        lo_cmp_api                   TYPE REF TO if_wd_component,
        lo_window_manager            TYPE REF TO if_wd_window_manager,
        lo_window                    TYPE REF TO if_wd_window.
  DATA: lv_host                      TYPE string,
        lv_port                      TYPE string,
        lv_out_protocol              TYPE string.

  CONSTANTS: lc_its_service_name     TYPE url_param
                                     VALUE '/sap/bc/gui/sap/its/webgui/',
             lc_sys_clnt             TYPE url_param VALUE '!?sap-client=',
             lc_its_transaction      TYPE url_param VALUE '&~transaction=',
             lc_web                  TYPE char04    VALUE '://',
             lc_vf_tran_field        TYPE url_param VALUE 'VF03&VBRK-VBELN=',
             lc_ok_code              TYPE char16    VALUE '&~okcode=%20',
             lc_colon                TYPE char01    VALUE ':'.

* When defining the event method the system automatically gives us a reference
* in the parameters of that method to the invoice number selected by the user.
* We can pass that value (lr_vbeln->*) to a local variable for our use

        lr_vbeln ?= r_param->value.
        lv_vbeln = lr_vbeln->*.

      * Get the host and protocol

        wd_comp_controller->execute_get_location(
          IMPORTING
            e_host         = lv_host
            e_out_protocol = lv_out_protocol
            e_port         = lv_port ).

        CONCATENATE lc_vf_tran_field
                    lv_vbeln
                    lc_ok_code
                    INTO lv_trans_code.

        CONCATENATE lv_out_protocol
                    lc_web
                    lv_host
                    lc_colon
                    lv_port
                    lc_its_service_name
                    lc_sys_clnt
                    lc_its_transaction
                    lv_trans_code
                    INTO
```

```
                  lv_url.
    *OPEN URL
      lo_cmp_api = wd_comp_controller->wd_get_api( ).

      IF lo_cmp_api IS NOT BOUND.
        RETURN.
      ENDIF.

      lo_window_manager = lo_cmp_api->get_window_manager( ).

      IF lo_window_manager IS NOT BOUND.
        RETURN.
      ENDIF.

      CALL METHOD lo_window_manager->create_external_window
        EXPORTING
          url          = lv_url
          is_resizable = abap_true
        RECEIVING
          window       = lo_window.

      lo_window->open( ).
```

In order to obtain the correct host, protocol and port number a call to a method in the assistance class that is called from my componentcontroller. This ensures that as my application moves through the system landscape the correct data is obtained.

```
  wd_comp_controller->execute_get_location(
    IMPORTING
      e_host         = lv_host
      e_out_protocol = lv_out_protocol
      e_port         = lv_port ).
```

Within my method of my assistance class I call an interface in class cl_http_server to procure my host, protocol and port number.

```
  CALL METHOD cl_http_server=>if_http_server~get_location
    IMPORTING
      host         = e_host
      port         = e_port
      out_protocol = e_out_protocol.
```

Related Content

Please include at least three references to SDN documents or web pages.

User Productivity

Web Dynpro for ABAP

Application Development and Composition

Copyright

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.