Web Dynpro ABAP: Dynamic Table

Applies to:

SAP ECC 6.0

Summary

Normally ABAP consultants might be aware of how to create internal table dynamically. This article aims to help the consultants how to display the dynamic table using Web Dynpro ABAP.

Author: J.Jayanthi

Company: Siemens Information Processing Services Pvt. Ltd.

Created on: 18.10.2010

Author Bio

J.Jayanthi is a ABAP Certified professional with HR ABAP Knowledge.

Table of Contents

Prerequisites 3
Goal 3
Creating Web Dynpro 3
Code 4
View 10
Window 10
Web Dynpro Application 10
Output 10
Related Content 11
Disclaimer and Liability Notice 12

Prerequisites

Have a look at the short definition about View and Window.

View

The view is the smallest unit of a Web Dynpro application visible for the user. The layout elements and dialog elements - for example, tables, text fields, or buttons - required for the application are arranged in a view. The view contains a controller and a controller context in which the application data to be processed is stored in a hierarchical structure. This allows the linking of the graphical elements with the application data.

Window

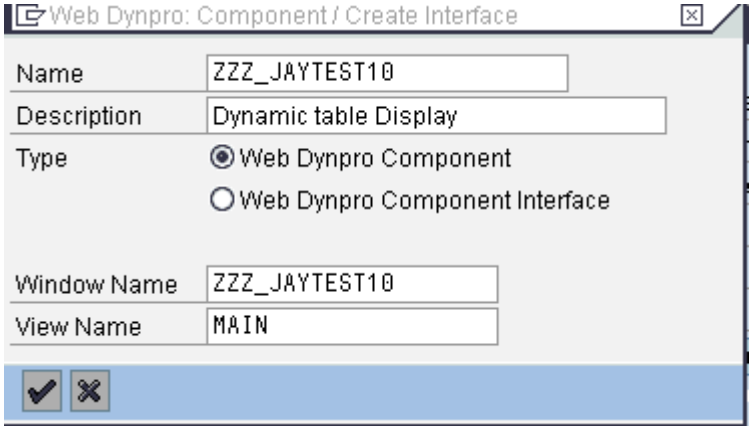
A window is used to group multiple views and to specify the navigation between the views. A view can only displayed by the browser if the view is embedded in a window.

Goal

In this article, we are going to display the document along with the number of items which will be displayed as material. For example, for some document, there can be 3 items. For another document, there can be 2 items. So for this document, 3rd item will be blank. Like that, whatever document has the maximum number of items in the given range of sales document that will be taken to decide the number of columns dynamically.

Creating Web Dynpro

Go to SE80 and select Web Dynpro Comp/Intf. and provide the name (say ZZZ_JAYTEST10) to create.
Then enter the description and choose the type as Web Dynpro Component.



Code

Since we are designing the table dynamically, coding plays a key role.

For Runtime services, there are few classes, in which we are going to use `cl_abap_structdescr` and `cl_abap_tabledescr`.

Here in this example, sales document and along with that the material numbers will be displayed. Find the document which has maximum item.

```
* Logic for data selection
  SELECT vbeln posnr matnr FROM vbap
  INTO TABLE t_vbap
  WHERE vbeln GE '1700000075' AND vbeln LE '1700000079'.

* Deciding the no. of columns for material
  LOOP AT t_vbap INTO wa_vbap.
    IF wa_vbap-vbeln = l_vbeln.
      ln = ln + 1.
    ELSE.
      IF ln > cnt.
        cnt = ln .
      ENDIF.
      ln = 1.
    ENDIF.
    l_vbeln = wa_vbap-vbeln.
  ENDLOOP.
```

Then decide the components of the table.

```
  DATA: comp_tab    TYPE cl_abap_structdescr=>component_table,
        comp        LIKE LINE OF comp_tab.
  * Building the components
  * Build VBELN as first column
    comp-name = 'VBELN'.
    comp-type ?= cl_abap_datadescr=>describe_by_name( 'VBELN_VA' ).
    APPEND comp TO comp_tab.

  * Build columns for no. of materials
    DO cnt TIMES.
      l_i = sy-tabix.
      CONCATENATE 'ITEM' l_i INTO l_fname.
      comp-name = l_fname.
      comp-type ?= cl_abap_datadescr=>describe_by_name( 'MATNR' ).
      APPEND comp TO comp_tab.
    ENDDO.
```

Next step is to build the structure using the components.

Use the methods GET_NODE_INFO of wd_context to get the node information and ADD_NEW_CHILD_NODE of `if_wd_context_node_info` to add Info object created at lower level.

To identify the low node, use get_child_node method of wd_context.

```
* Building Dynamic Structure
  struct_type = cl_abap_structdescr=>create( comp_tab ).

* Returns Meta Information About Node
  CALL METHOD wd_context->get_node_info
    RECEIVING
      node_info = node_info.
* Creates a Info Object and Adds it as a Lower-Level Node
  CALL METHOD node_info->add_new_child_node
    EXPORTING
      name                = 'MY_NODE'
      static_element_rtti = struct_type
      is_static           = abap_false
    RECEIVING
      child_node_info     = node_info.

* Child Node of Lead Selection or Specific Index
  CALL METHOD wd_context->get_child_node
    EXPORTING
      name       = 'MY_NODE'
    RECEIVING
      child_node = node.
* Returns RTTI Object for Structure Type of Static Attributes
  CALL METHOD node_info->get_static_attributes_type
    RECEIVING
      rtti = struct_type.
```

Now create the table and work area.

```
* Create table
  table_type = cl_abap_tabledescr=>create( p_line_type = struct_type ).
  CREATE DATA my_table TYPE HANDLE table_type.
  ASSIGN my_table->* TO <table>.
* Create lines
  CREATE DATA my_row TYPE HANDLE struct_type.
  ASSIGN my_row->* TO <row>.
```

Populate the values.

```
  ASSIGN COMPONENT 'VBELN' OF STRUCTURE <row> TO <fs_vbeln>.
* Populate temp table
  t_temp[] = t_vbap[].
  LOOP AT t_vbap INTO wa_vbap.
    <fs_vbeln> = wa_vbap-vbeln.
    CLEAR : wa_temp, ln.
    LOOP AT t_temp INTO wa_temp WHERE vbeln = wa_vbap-vbeln.
      l_i = sy-tabix.
      CONCATENATE 'ITEM' l_i INTO l_fname.
      ASSIGN COMPONENT  l_fname OF STRUCTURE <row> TO <fs_item>.
      <fs_item> = wa_temp-matnr.
      ln = ln + 1.
    ENDLOOP.
    IF ln LT cnt.
      DO.
        ln = ln + 1.
        l_i = ln.
        CONCATENATE 'ITEM' l_i INTO l_fname.
        ASSIGN COMPONENT  l_fname OF STRUCTURE <row> TO <fs_item>.
        <fs_item> = ' '.
        IF ln EQ cnt.
          EXIT.
        ENDIF.
      ENDDO.
    ENDIF.
    APPEND <row> TO <table>.
    DELETE t_vbap WHERE vbeln = wa_vbap-vbeln.
    DELETE t_temp WHERE vbeln = wa_vbap-vbeln.
  ENDLOOP.
```

To bind the table

```
* Bind the table
  node->bind_table( <table> ).
```

Then in WDDOMODIFYVIEW method, do the following.

CL_WD_DYNAMIC_TOOL should be used to create table from node. If it is not written, output will be empty.

```
method WDDOMODIFYVIEW .
  data: l_root type ref to cl_wd_uielement_container,
        l_node type ref to if_wd_context_node,
         l_table TYPE REF TO cl_wd_table.
  if first_time = abap_true.
    l_root ?= view->GET_ROOT_ELEMENT( ).
    l_node = wd_context->get_child_node( 'MY_NODE' ).

    cl_wd_dynamic_tool=>create_table_from_node(
    ui_parent = l_root
    table_id = 'MY_TABLE'
    node = l_node ).
  endif.
endmethod.
```

Complete code is as below.

```
METHOD wddoinit .
types : begin of ty_vbap,
        vbeln type vbap-vbeln,
        posnr type vbap-posnr,
        matnr type vbap-matnr,
        end of ty_vbap.
  DATA: comp_tab    TYPE cl_abap_structdescr=>component_table,
        comp        LIKE LINE OF comp_tab,
        struct_type TYPE REF TO cl_abap_structdescr,
        table_type  TYPE REF TO cl_abap_tabledescr,
        node_info TYPE REF TO if_wd_context_node_info,
        node TYPE REF TO if_wd_context_node,
        my_table    TYPE REF TO data,
        my_row      TYPE REF TO data,
         l_i TYPE c,
         l_fname(6),
         l_vbeln TYPE vbap-vbeln,
         t_vbap TYPE STANDARD TABLE OF ty_vbap,
         wa_vbap TYPE ty_vbap,
         t_temp TYPE STANDARD TABLE OF ty_vbap,
         wa_temp TYPE ty_vbap,
         ln TYPE i VALUE 1,
         cnt TYPE i VALUE 1.

  FIELD-SYMBOLS: <table> TYPE table,
                 <row> TYPE data,
                 <fs_vbeln> TYPE vbap-vbeln,
                 <fs_item> TYPE vbap-matnr.
```

```
* Logic for data selection
  SELECT vbeln posnr matnr FROM vbap
  INTO TABLE t_vbap
  WHERE vbeln GE '1700000075' AND vbeln LE '1700000079'.
* Deciding the no. of columns for material
  LOOP AT t_vbap INTO wa_vbap.
    IF wa_vbap-vbeln = l_vbeln.
      ln = ln + 1.
    ELSE.
      IF ln > cnt.
        cnt = ln .
      ENDIF.
      ln = 1.
    ENDIF.
    l_vbeln = wa_vbap-vbeln.
  ENDLOOP.
* Building the components
* Build VBELN as first column
  comp-name = 'VBELN'.
  comp-type ?= cl_abap_datadescr=>describe_by_name( 'VBELN_VA' ).
  APPEND comp TO comp_tab.
```

```
* Build columns for no. of materials
  DO cnt TIMES.
    l_i = sy-tabix.
    CONCATENATE 'ITEM' l_i INTO l_fname.
    comp-name = l_fname.
    comp-type ?= cl_abap_datadescr=>describe_by_name( 'MATNR' ).
    APPEND comp TO comp_tab.
  ENDDO.

* Building Dynamic Structure
  struct_type = cl_abap_structdescr=>create( comp_tab ).

* Returns Meta Information About Node
  CALL METHOD wd_context->get_node_info
    RECEIVING
      node_info = node_info.
          _                 _
* Creates a Info Object and Adds it as a Lower-Level Node
  CALL METHOD node_info->add_new_child_node
    EXPORTING
      name                = 'MY_NODE'
      static_element_rtti = struct_type
      is_static           = abap_false
    RECEIVING
      child_node_info     = node_info.

* Child Node of Lead Selection or Specific Index
  CALL METHOD wd_context->get_child_node
    EXPORTING
      name       = 'MY_NODE'
    RECEIVING
      child_node = node.
* Returns RTTI Object for Structure Type of Static Attributes
  CALL METHOD node_info->get_static_attributes_type
    RECEIVING
      rtti = struct_type.
* Create table
  table_type = cl_abap_tabledescr=>create( p_line_type = struct_type ).
  CREATE DATA my_table TYPE HANDLE table_type.
  ASSIGN my_table->* TO <table>.
```

```
* Create lines
  CREATE DATA my_row TYPE HANDLE struct_type.
  ASSIGN my_row->* TO <row>.

* Populate the values for table
  ASSIGN COMPONENT 'VBELN' OF STRUCTURE <row> TO <fs_vbeln>.
* Populate temp table
  t_temp[] = t_vbap[].
  LOOP AT t_vbap INTO wa_vbap.
    <fs_vbeln> = wa_vbap-vbeln.
    CLEAR : wa_temp, ln.
    LOOP AT t_temp INTO wa_temp WHERE vbeln = wa_vbap-vbeln.
      l_i = sy-tabix.
      CONCATENATE 'ITEM' l_i INTO l_fname.
      ASSIGN COMPONENT  l_fname OF STRUCTURE <row> TO <fs_item>.
      <fs_item> = wa_temp-matnr.
      ln = ln + 1.
    ENDLOOP.
```

```
    IF ln LT cnt.
      DO.
        ln = ln + 1.
        l_i = ln.
        CONCATENATE 'ITEM' l_i INTO l_fname.
        ASSIGN COMPONENT  l_fname OF STRUCTURE <row> TO <fs_item>.
        <fs_item> = ' '.
        IF ln EQ cnt.
          EXIT.
        ENDIF.
      ENDDO.
    ENDIF.
    APPEND <row> TO <table>.
    DELETE t_vbap WHERE vbeln = wa_vbap-vbeln.
    DELETE t_temp WHERE vbeln = wa_vbap-vbeln.
  ENDLOOP.
* Bind the table
  node->bind_table( <table> ).
ENDMETHOD.
```

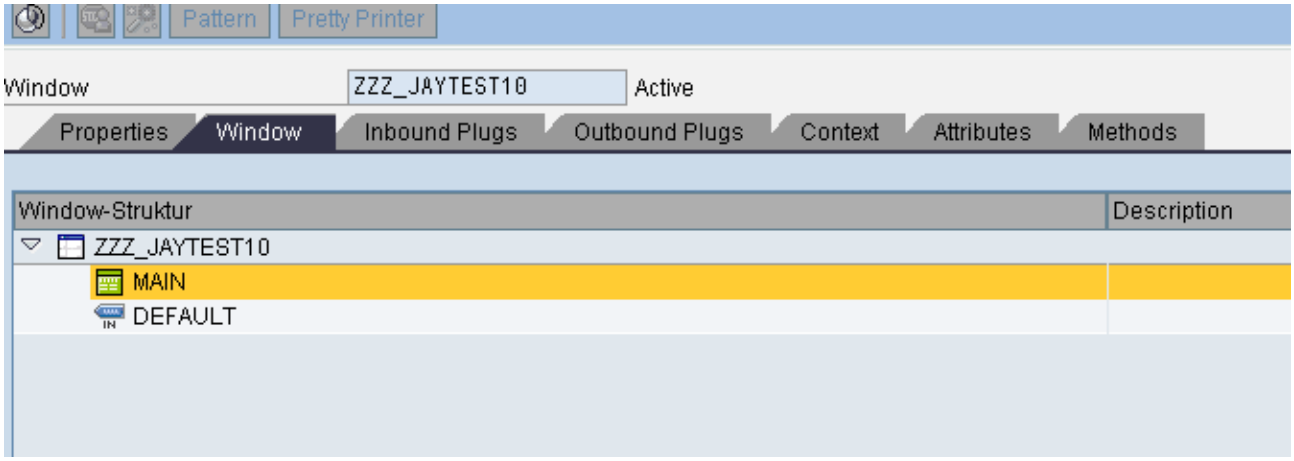
```
method WDDOMODIFYVIEW .
  data: l_root type ref to cl_wd_uielement_container,
        l_node type ref to if_wd_context_node,
         l_table TYPE REF TO cl_wd_table.
  if first_time = abap_true.
    l_root ?= view->GET_ROOT_ELEMENT( ).
    l_node = wd_context->get_child_node( 'MY_NODE' ).

    cl_wd_dynamic_tool=>create_table_from_node(
    ui_parent = l_root
    table_id = 'MY_TABLE'
    node = l_node ).
  endif.
endmethod.
```

View

Double click the view (Main). There is no need to do anything in Layout since we are going to design table dynamically.

Window

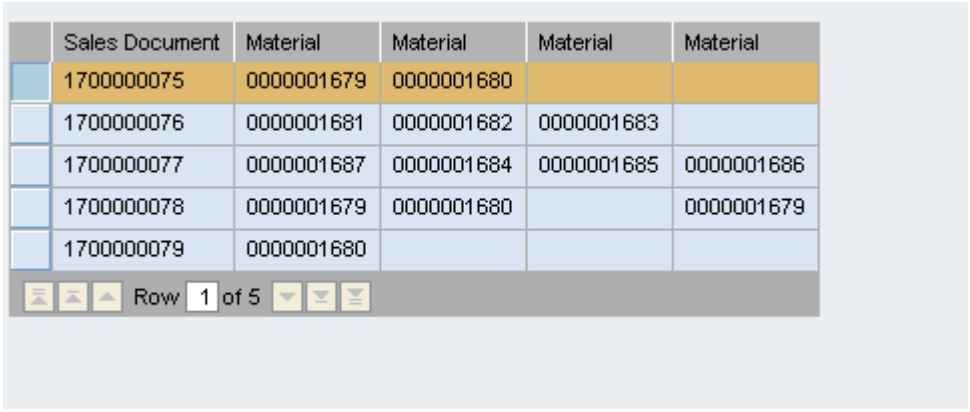
In the default window, embed the view as below.



Web Dynpro Application

Create Web Dynpro Application by right clicking the object (ZZZ_JAYTEST10).

Output



Sales Document	Material	Material	Material	Material
1700000075	0000001679	0000001680		
1700000076	0000001681	0000001682	0000001683	
1700000077	0000001687	0000001684	0000001685	0000001686
1700000078	0000001679	0000001680		0000001679
1700000079	0000001680			

Row 1 of 5

Related Content

Web Dynpro: Coloring Table Conditionally

Web Dynpro: Column Coloring in ALV

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.