Web Dynpro ABAP: Changing ALV Contents and Saving in Database

Applies to:

SAP ECC 6.0. For more information, visit the Web Dynpro ABAP homepage

Summary

The article is aimed to help beginners in Webdynpro ABAP who wants to change the ALV and save the changes in Database.

Author: J.Jayanthi

Company: Siemens Information Processing Services Pvt. Ltd.

Created on: November 4, 2010

Author Bio

J.Jayanthi is a Certified ABAP consultant with HR ABAP knowledge.

Table of Contents

ALV in Web Dynpro ABAP....3
Prerequisites....3
Creating Web Dynpro....4
Component Controller....4
Component Usages....5
Designing View....5
Embedding View....11
Creating Web Dynpro Application....12
Code....12
Output……....15
Related Content....16
Disclaimer and Liability Notice....17

ALV in Web Dynpro ABAP

ABAP Consultants normally do changes in ALV and save the changes in Database. This article explains how to achieve this task in Webdynpro ABAP.

Prerequisites

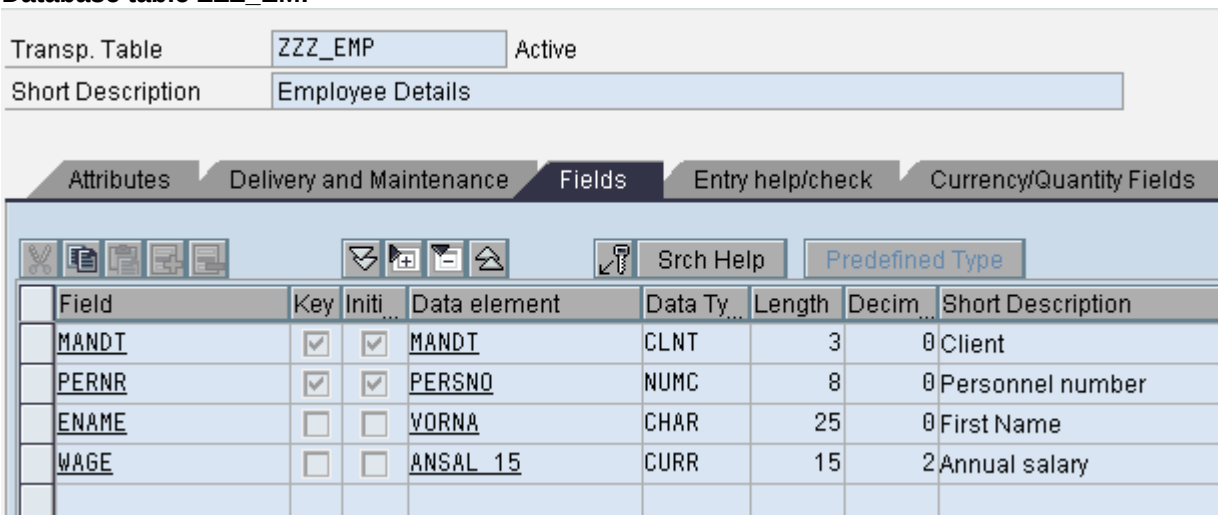
Component
The component is the central, reusable unit of the application project. You can create any number of views in a component and arrange them in any number of windows.

Component Usages
Web Dynpro components can be nested. This means that you can integrate any number of other, already existing components into a component.

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.

Database table ZZZ_EMP

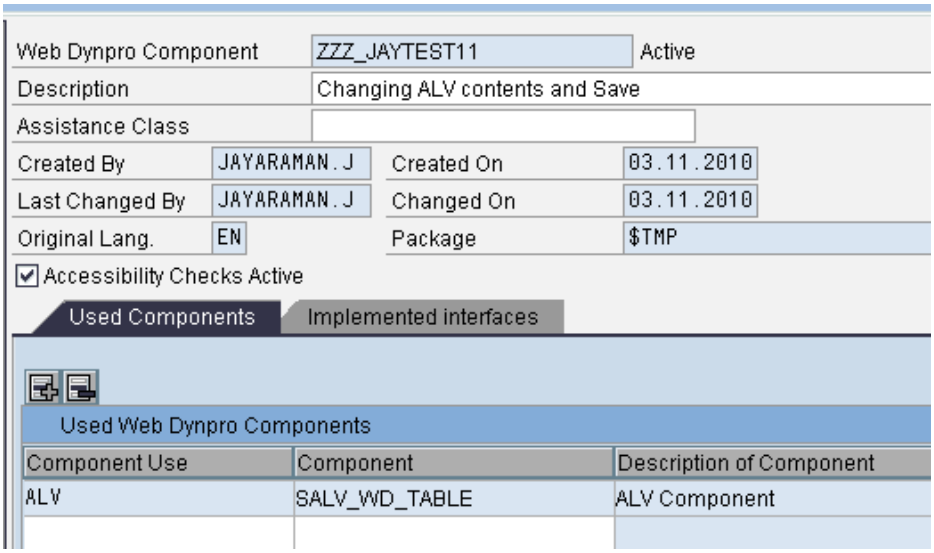


Field	Key	Initi	Data element	Data Ty	Length	Decim	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	Client
PERNR	☑	☑	PERSNO	NUMC	8	0	Personnel number
ENAME	☐	☐	VORNA	CHAR	25	0	First Name
WAGE	☐	☐	ANSAL_15	CURR	15	2	Annual salary

Creating Web Dynpro

Go to SE80 and select Web Dynpro Comp./Intf. and provide the name(say ZZZ_JAYTEST11) and create. Enter the description and choose the type as Web Dynpro Component.

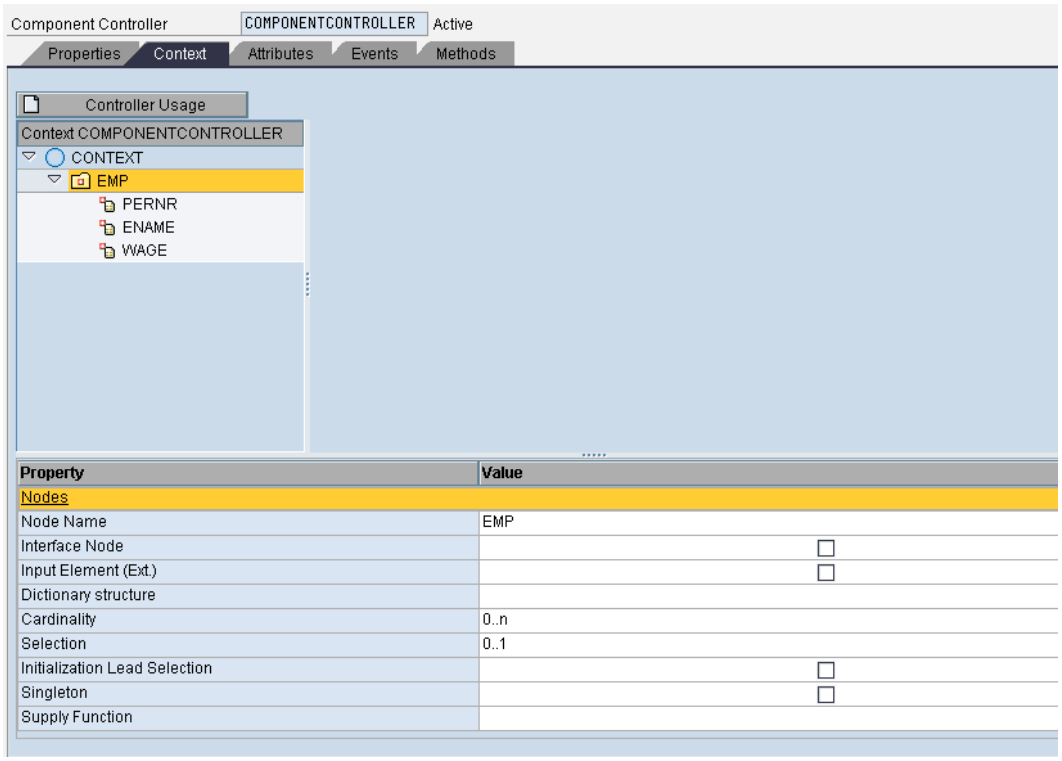
Mention the Component Use as ALV and Component as SALV_WD_TABLE in the Used Components tab in Web Dynpro (ZZZ_JAYTEST11).



This will create a Component Usages by name ALV.

Component Controller

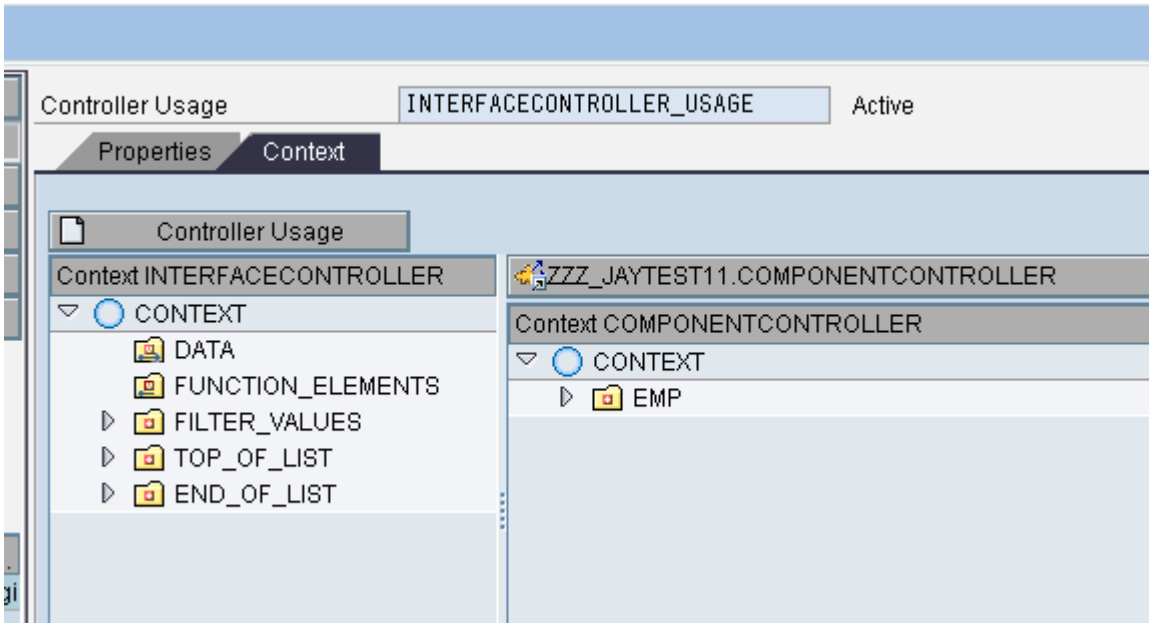
Go to Component Controller and Right click the context. Then select Create Node EMP with dictionary structure ZZZ_EMP and select the all the attributes except client from ZZZ_EMP. Then remove the table name from the dictionary structure in EMP node and set the properties as below.



Component Usages

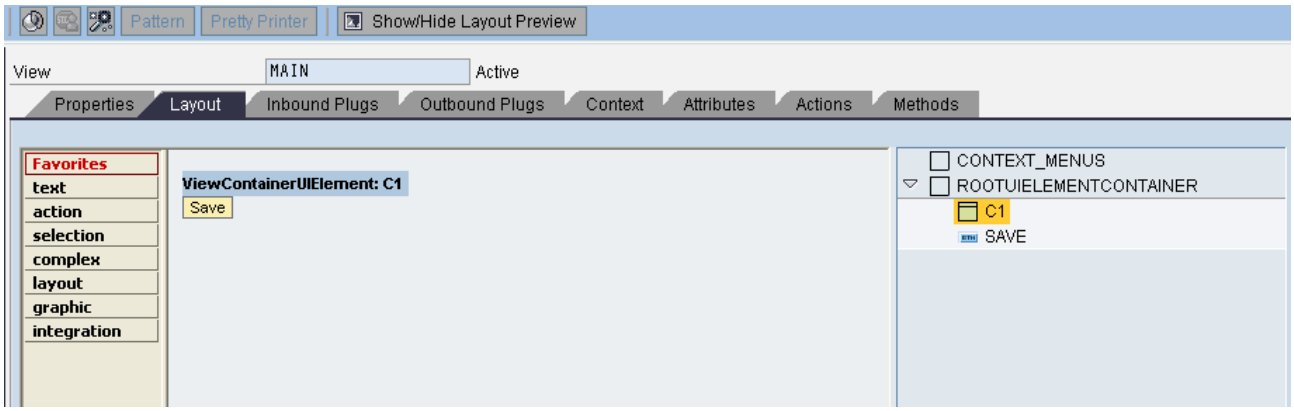
Right click the component Usage (here with name ALV) and Create controller Usage.

Drag and drop the node(EMP in right side) from Component Controller context to Data(in left side) in Controller Usage Context.

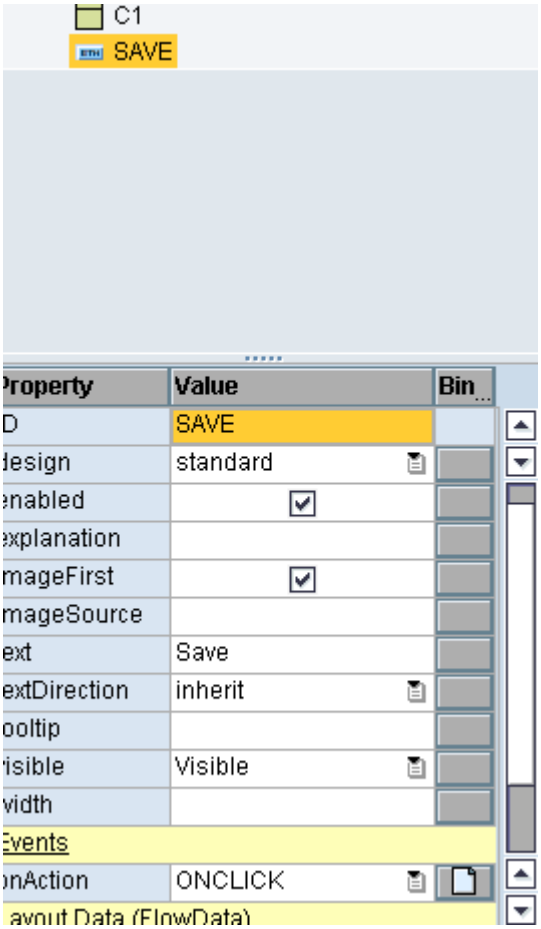


Designing View

Go to the layout in view and right click the ROOTUIELEMENTCONTAINER and then choose Insert element.Fill the ID and choose the Typ as ViewContainerUIElement and then Insert element Button by name SAVE. After doing the same, the layout will appear as below.

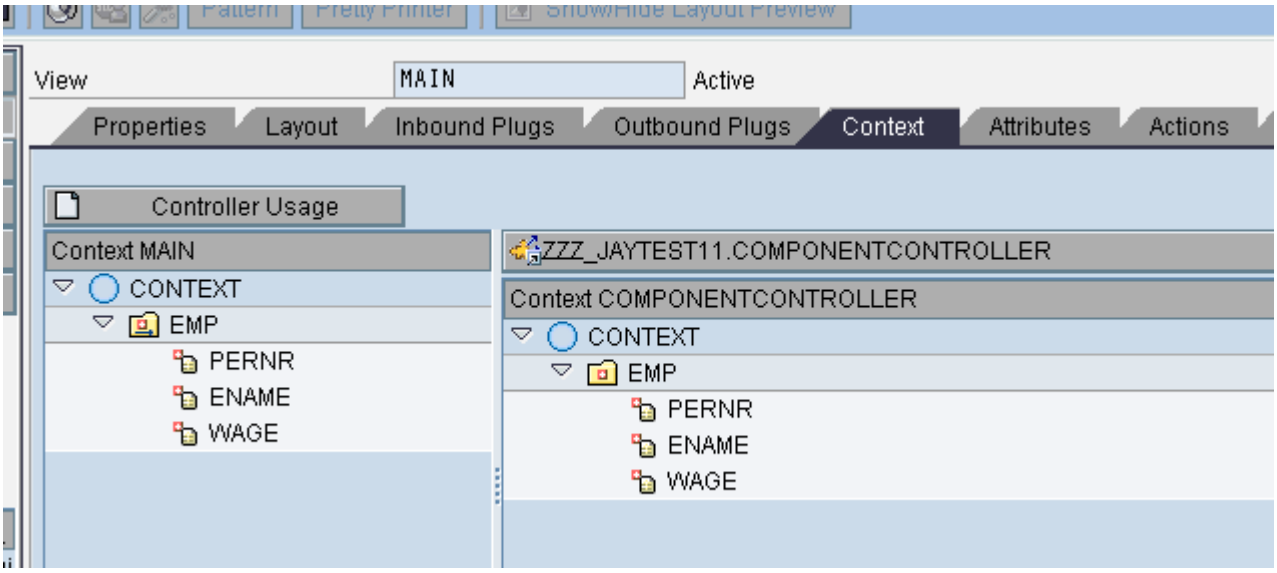


Create action ONCLICK for save button.

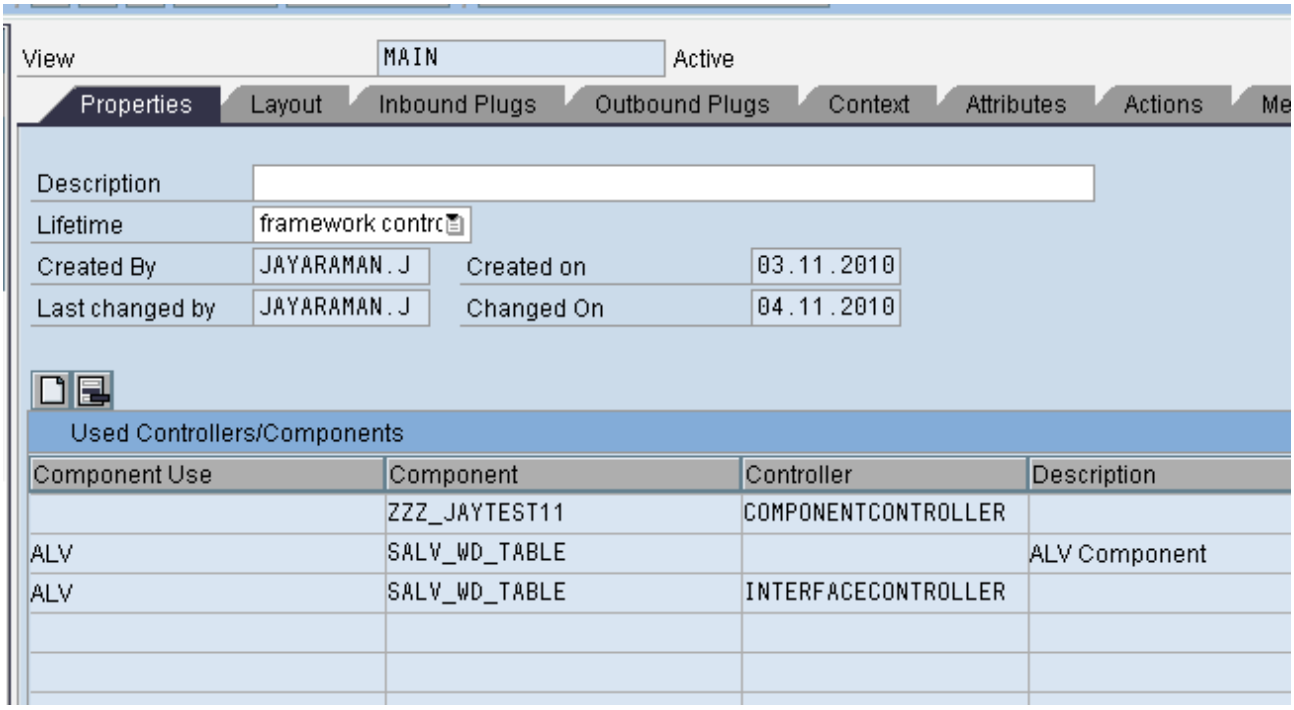


In context tab, drag and drop the context which appears under component controller to view(Main is the view name). After drag and drop, the context will appear as below.

 symbol in left side shows it is mapped.



In the properties, define as below.



Select the method WDDOINIT in methods tab.

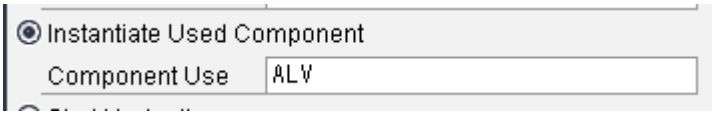
Use Web Dynpro code wizard  to generate code automatically.

Step a: Choose the radio button Read context and then press F4 to select the context EMP.

Keep the below generated code and delete the rest which is not required.

```
* Reading the context node emp
  DATA lo_nd_emp TYPE REF TO if_wd_context_node.
* navigate from <CONTEXT> to <EMP> via lead selection
  lo_nd_emp = wd_context->get_child_node( name = wd_this->wdctx_emp ).
```

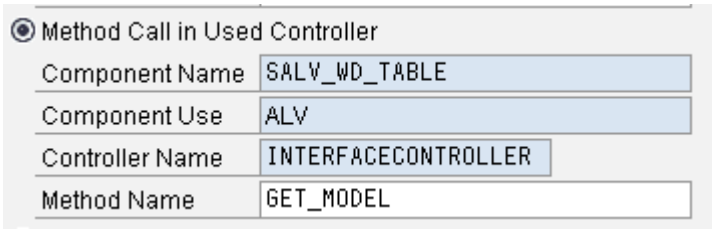
Step b: Again use Code Wizard as below.



This will generate the below code.

```
  DATA lo_cmp_usage TYPE REF TO if_wd_component_usage.
  lo_cmp_usage =   wd_this->wd_cpuse_alv( ).
  IF lo_cmp_usage->has_active_component( ) IS INITIAL.
    lo_cmp_usage->create_component( ).
  ENDIF.
```

Step c: Then use Method call in Used controller as below in Code Wizard.




```
* Get config model
  DATA lo_interfacecontroller TYPE REF TO iwci_salv_wd_table .
  lo_interfacecontroller =   wd_this->wd_cpifc_alv( ).

  DATA lo_value TYPE REF TO cl_salv_wd_config_table.
  lo_value = lo_interfacecontroller->get_model( ).
```

Step d: Then select data using logic and bind table

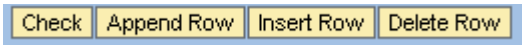
```
* Selection logic
  data t_emp type standard table of zzz_emp.
  select * from zzz_emp into table t_emp.
```

```
  lo_nd_emp->bind_table( t_emp ).
```

Step e: To make the alv appear with editable toolbar and set read only mode, do the following.

```
data lr_table_settings TYPE REF TO if_salv_wd_table_settings.
* set read only mode to false (and display edit toolbar)
  lr_table_settings ?= lo_value.
  lr_table_settings->set_read_only( abap_false ).
```

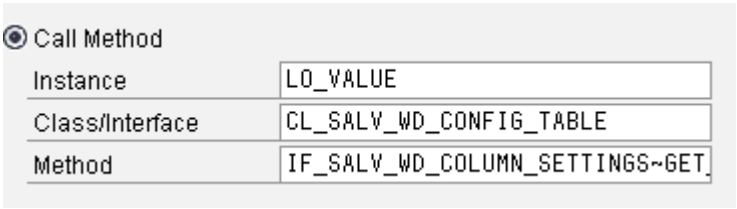
Which in turn makes the following appear in the toolbar.



To hide these buttons, do the following.

```
* To disable toolbar buttons
lo_value->if_salv_wd_std_functions~SET_EDIT_APPEND_ROW_ALLOWED(
abap_false ).
lo_value->if_salv_wd_std_functions~SET_EDIT_CHECK_AVAILABLE(
abap_false ).
lo_value->if_salv_wd_std_functions~SET_EDIT_INSERT_ROW_ALLOWED(
abap_false ).
lo_value->if_salv_wd_std_functions~SET_EDIT_DELETE_ROW_ALLOWED(
abap_false ).
```

Step f: Retrieve the columns as below.

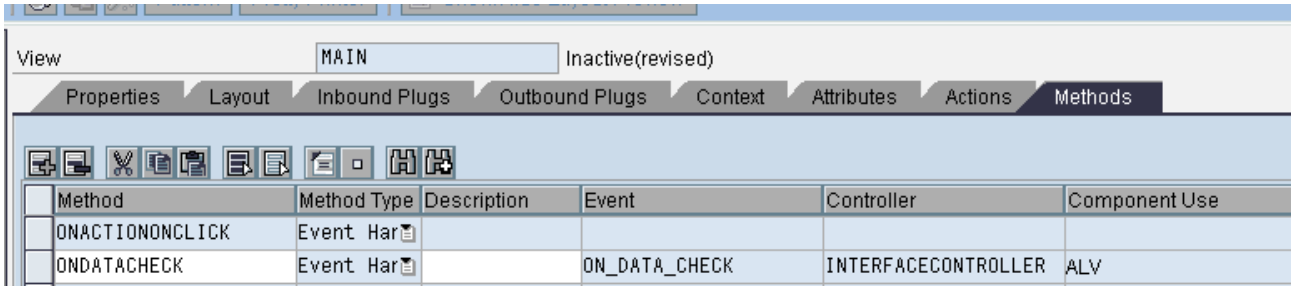



```
  DATA : lr_columns TYPE salv_wd_t_column_ref,
         wa_cols TYPE salv_wd_s_column_ref.
* Returns the Objects of All Columns
CALL METHOD lo_value->if_salv_wd_column_settings~get_columns
  receiving
    value  = lr_columns.
```

Step g: Use the CL_SALV_WD_UIE_INPUT_FIELD class to make the particular column as editable. Here we are going to make the field Wage as editable.

```
* With the CL_SALV_WD_UIE_INPUT_FIELD class, we can define all the
*settings for the input field UI element, with which you can make
*entries to the ALV output.
  DATA lr_input_field TYPE REF TO cl_salv_wd_uie_input_field.

  LOOP AT lr_columns INTO wa_cols.
    CREATE OBJECT lr_input_field
      EXPORTING
        value_fieldname = wa_cols-id.
    CASE wa_cols-id.
      WHEN 'WAGE'.
        CALL METHOD wa_cols-r_column->set_cell_editor
          EXPORTING
            value = lr_input_field.
    ENDCASE.
    FREE lr_input_field.
  ENDLOOP.
```

Now we need to do the coding for SAVE Method. Before that we need to add the Event Handler for datacheck.



Now we need to do coding in Save method. First Call the method WD_CPIFC_ALV in current controller.

This will generate the below code.

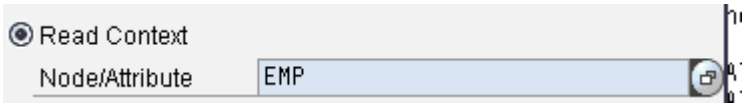
```
  DATA lo_interface TYPE REF TO iwci_salv_wd_table.
  lo_interface = wd_this->wd_cpifc_alv( ).
```

Then call the DATACHECK method.

```
* Check for changes
  lo_interface->data_check( ).
```

Now we are going to write the logic in data_check for capturing the modified values and updating the database.

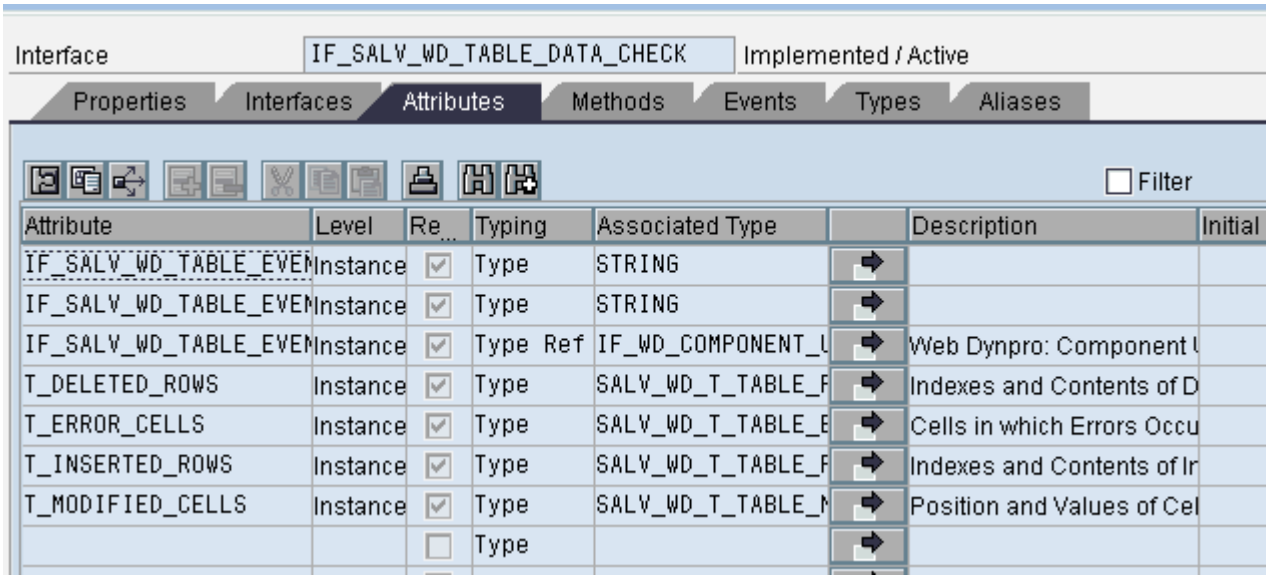
Step a: Read the context node EMP.



```
  DATA lo_nd_emp TYPE REF TO if_wd_context_node.
* navigate from <CONTEXT> to <EMP> via lead selection
  lo_nd_emp = wd_context->get_child_node( name = wd_this->wdctx_emp ).
```

Step b:

R_param which is the import parameter has the following attributes.



Identify the modified values.

```
  data : t_mod type standard table of zzz_emp,
         wa_mod type zzz_emp,
         t_table TYPE if_main=>elements_emp,
         wa_table TYPE if_main=>element_emp,
         wa_param type SALV_WD_S_TABLE_MOD_CELL.
* Check whether there is any modified data
  check r_param->t_modified_cells is not initial.
  lo_nd_emp->get_static_attributes_table( IMPORTING table = t_table ).
  loop at r_param->t_modified_cells into wa_param.
* Retrieve the data from the t_table using the index
    read table t_table into wa_table index wa_param-index.
    if sy-subrc eq 0.
       move-corresponding wa_table to wa_mod.
       append wa_mod to t_mod.
    endif.
  endloop.
  MODIFY zzz_emp FROM TABLE t_mod.
```

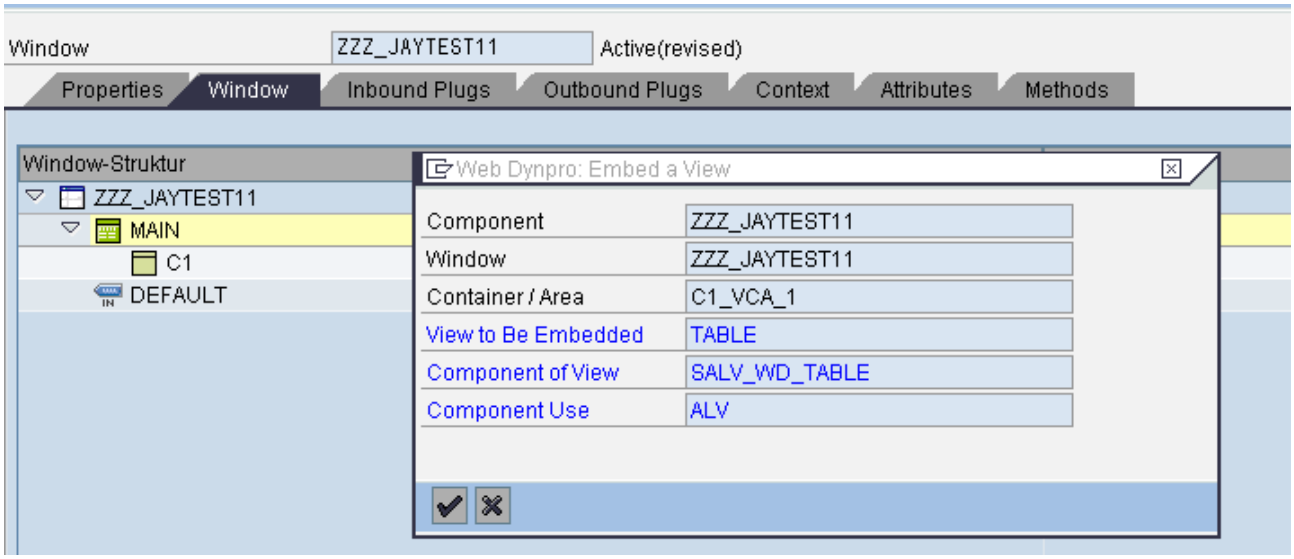
Step c:

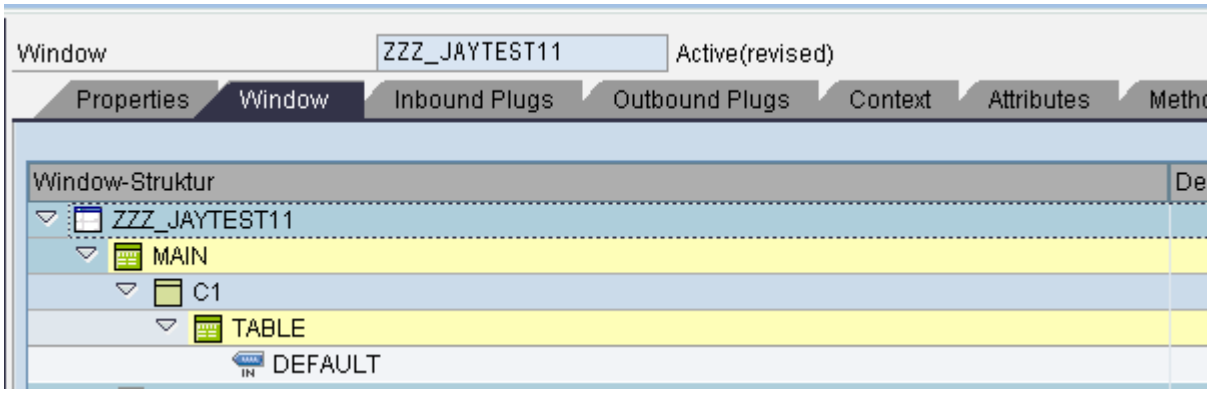
Use message manager to populate the message.

```
* Populating message
data : lo_api_controller     TYPE REF TO if_wd_controller,
       lo_message_manager    TYPE REF TO if_wd_message_manager.

  lo_api_controller ?= wd_this->wd_get_api( ).
* Get message manager
  CALL METHOD lo_api_controller->get_message_manager
    RECEIVING
      message_manager = lo_message_manager.
  CALL METHOD lo_message_manager->report_success
    EXPORTING
      message_text = 'Saved successfully'.
```

Embedding View

Embed the view by right clicking the C1 in the window.



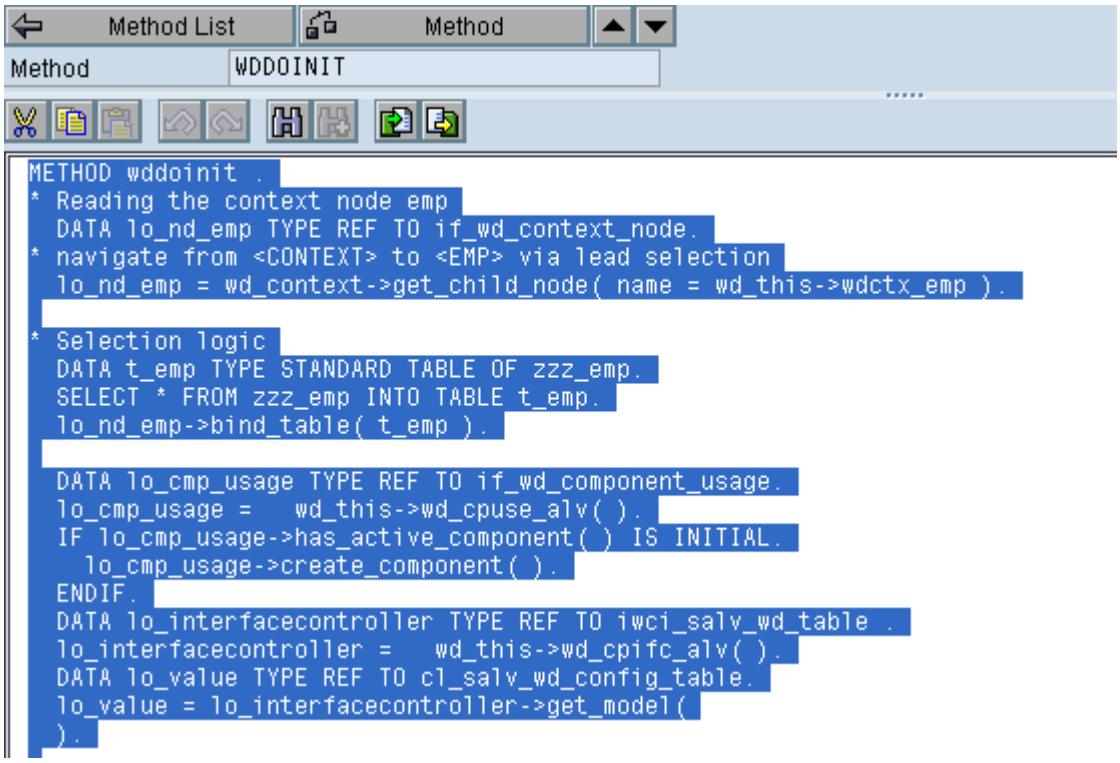


Creating Web Dynpro Application

Create Web Dynpro Application by right clicking the Webdynpro(ZZZ_JAYTEST11).

Right click the Web Dynpro component and activate.

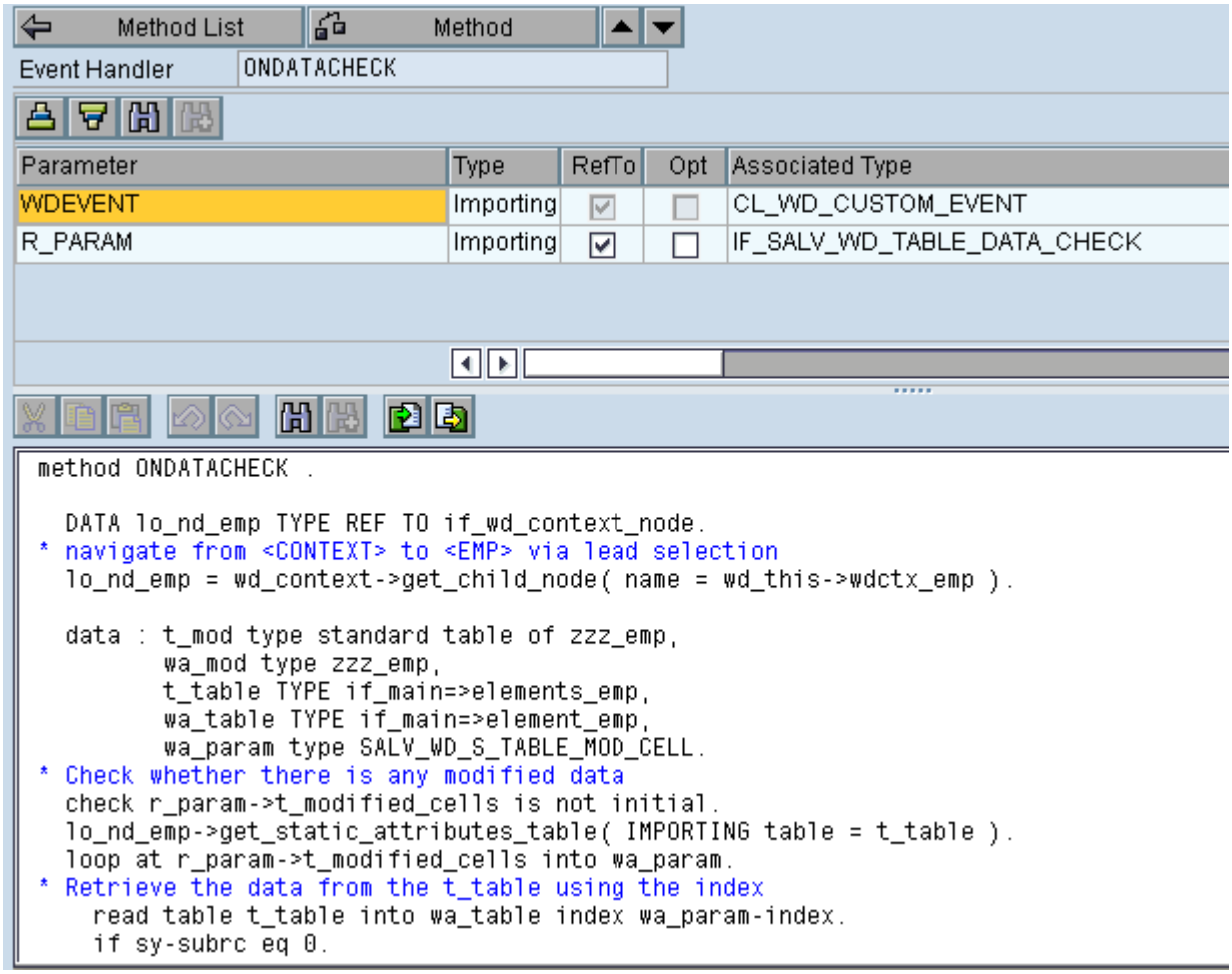
Code



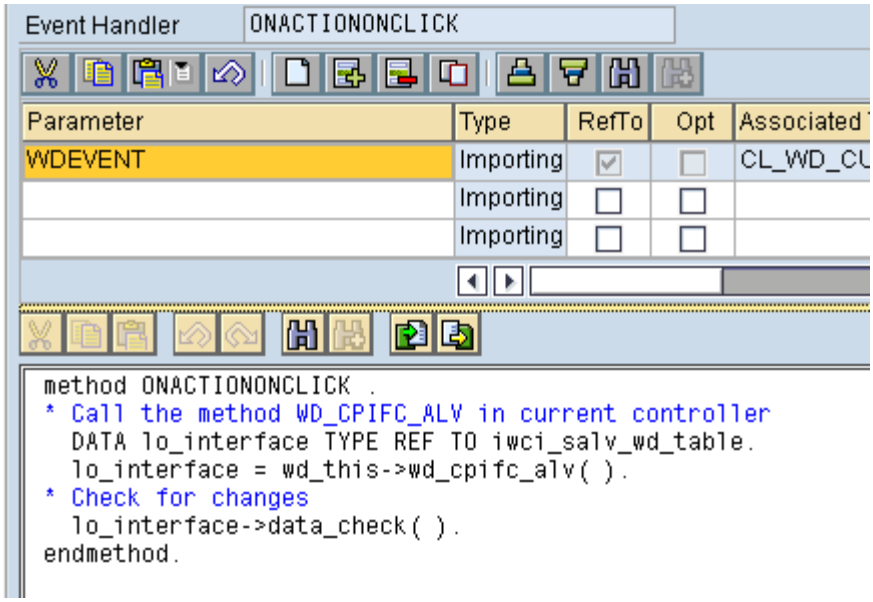

```
METHOD wddoinit .
* Reading the context node emp
  DATA lo_nd_emp TYPE REF TO if_wd_context_node.
* navigate from <CONTEXT> to <EMP> via lead selection
  lo_nd_emp = wd_context->get_child_node( name = wd_this->wdctx_emp ).

* Selection logic
  DATA t_emp TYPE STANDARD TABLE OF zzz_emp.
  SELECT * FROM zzz_emp INTO TABLE t_emp.
  lo_nd_emp->bind_table( t_emp ).

  DATA lo_cmp_usage TYPE REF TO if_wd_component_usage.
  lo_cmp_usage =   wd_this->wd_cpuse_alv( ).
  IF lo_cmp_usage->has_active_component( ) IS INITIAL.
    lo_cmp_usage->create_component( ).
  ENDIF.
  DATA lo_interfacecontroller TYPE REF TO iwci_salv_wd_table .
  lo_interfacecontroller =   wd_this->wd_cpifc_alv( ).
  DATA lo_value TYPE REF TO cl_salv_wd_config_table.
  lo_value = lo_interfacecontroller->get_model(
  ).
```

```
  DATA lr_table_settings TYPE REF TO if_salv_wd_table_settings.
* set read only mode to false (and display edit toolbar)
  lr_table_settings ?= lo_value.
  lr_table_settings->set_read_only( abap_false ).
* To disable toolbar buttons
  lo_value->if_salv_wd_std_functions~set_edit_append_row_allowed(
  abap_false ).
  lo_value->if_salv_wd_std_functions~set_edit_check_available(
  abap_false ).
  lo_value->if_salv_wd_std_functions~set_edit_insert_row_allowed(
  abap_false ).
  lo_value->if_salv_wd_std_functions~set_edit_delete_row_allowed(
  abap_false ).

  DATA : lr_columns TYPE salv_wd_t_column_ref,
         wa_cols TYPE salv_wd_s_column_ref.
* Returns the Objects of All Columns
CALL METHOD lo_value->if_salv_wd_column_settings~get_columns
  receiving
    value  = lr_columns
```

```
* With the CL_SALV_WD_UIE_INPUT_FIELD class, we can define all the
*settings for the input field UI element, with which you can make
*entries to the ALV output.
  DATA lr_input_field TYPE REF TO cl_salv_wd_uie_input_field.

  LOOP AT lr_columns INTO wa_cols.
    CREATE OBJECT lr_input_field
      EXPORTING
        value_fieldname = wa_cols-id.
    CASE wa_cols-id.
      WHEN 'WAGE'.
        CALL METHOD wa_cols-r_column->set_cell_editor
          EXPORTING
            value = lr_input_field.
    ENDCASE.
    FREE lr_input_field.
  ENDLOOP.

ENDMETHOD.
```

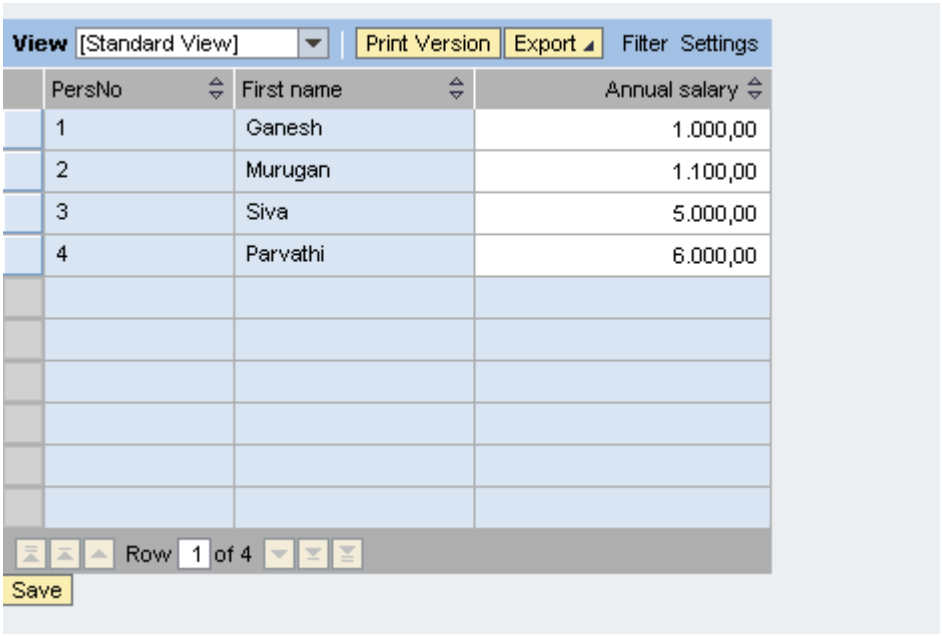


```
        move-corresponding wa_table to wa_mod.
        append wa_mod to t_mod.
      endif.
    endloop.
    MODIFY zzz_emp FROM TABLE t_mod.
    if sy-subrc eq 0.
* Populating message
data : lo_api_controller     TYPE REF TO if_wd_controller,
       lo_message_manager    TYPE REF TO if_wd_message_manager.

    lo_api_controller ?= wd_this->wd_get_api( ).
* Get message manager
    CALL METHOD lo_api_controller->get_message_manager
      RECEIVING
        message_manager = lo_message_manager.
    CALL METHOD lo_message_manager->report_success
      EXPORTING
        message_text = 'Saved successfully'.
endif.
endmethod.
```

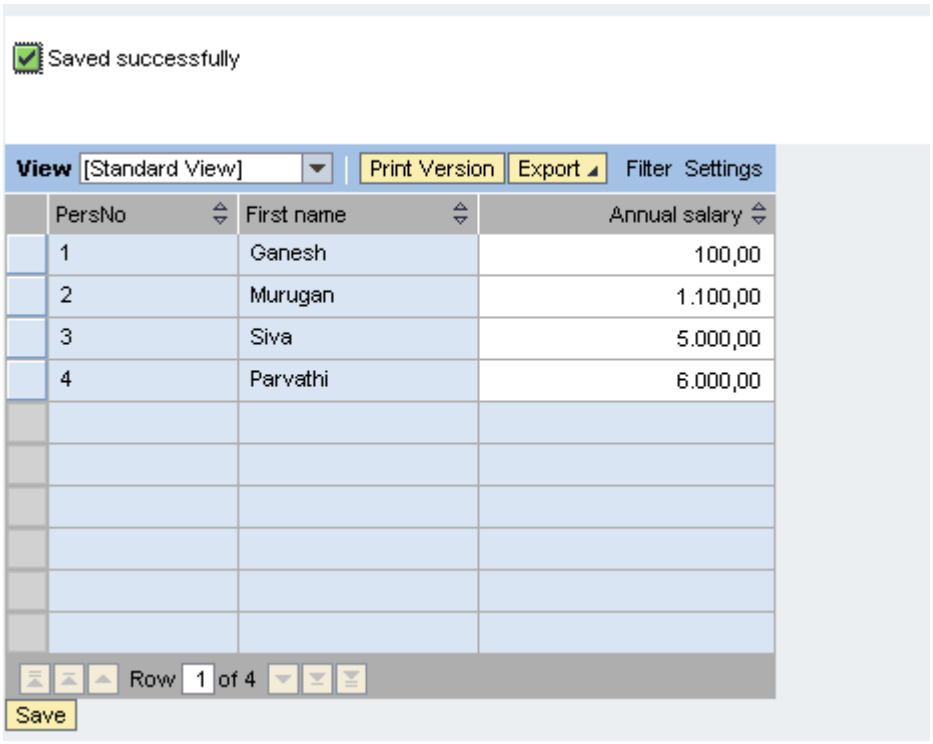


Output

Only the field Wage(Annual Salary) is editable.



After changing the first row wage from 1000 to 100, the change is getting reflected and message is shown for successful update.



Related Content

Reference 1

Reference 2

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.