Editing ALV in Web Dynpro for ABAP

Release SAP NetWeaver 2004s

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example text	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

Table of Contents

Handling of Editable ALV Tables in Web Dynpro

This tutorial shows you how to make a column of your ALV table editable.

Task

The starting point of this tutorial is the solution application of the tutorial, “Programming the ALV Configuration Model in Web Dynpro for ABAP.”

The task of this tutorial is to configure the ALV to make column “price” editable.

Objectives

By the end of this tutorial, you will be able to:

- ✓ Make a column of the ALV table editable

Knowledge

- ☐ Knowledge of ABAP OO programming language
- ☐ Basic knowledge of programming Web Dynpro applications
- ☐ Basic knowledge of ABAP workbench
- ☐ Knowledge of the tutorial “Simple example for using the ALV inside Web Dynpro for ABAP”.
- ☐ Knowledge of the tutorial “Programming the ALV configuration model inside Web Dynpro for ABAP”

Copying an Existing Web Dynpro Component

In the system there is a master copy of a Web Dynpro component called ***WDT_FLIGHTLIST_CONFIG***. You can copy this component as described below.

Procedure

Copying the Web Dynpro Component

1. Start the ABAP Workbench (*se80*) and select the Web Dynpro component *WDT_FLIGHTLIST_CONFIG*.
2. Open the context menu of *WDT_FLIGHTLIST_CONFIG* and copy the Web Dynpro component to Z00*_WDT_FLIGHTLIST_EDIT*.
3. Open the context menu of the new component Z00*_WDT_FLIGHTLIST_EDIT* and create a Web Dynpro application Z00*_WDT_FLIGHTLIST_EDIT.*
4. Select the interface view by using F4 help. Choose *MAIN.*
5. Select a plug name by using F4 help and choose *default*.
6. Activate the new Web Dynpro component.

Modify View ResultView for Displaying ALV Table

In this tutorial we are changing the ALV table from tutorial “Programming the ALV Configuration Model in Web Dynpro for ABAP” by making column “price” editable.

Configure ALV in ResultView to make ALV editable.

Set table editable.

First it is necessary to use an input field as cell editor for the column “price,” which should be editable. Therefore we need to enhance method WDDOINIT of view ResultView:

WDDOINIT()

```
[…]

* set cell editor for input fields (~make colum PRICE editable)
  DATA: lr_column_settings TYPE REF TO if_salv_wd_column_settings,
        lr_input_field     TYPE REF TO cl_salv_wd_uie_input_field.


  lr_column_settings ?= l_value.
  lr_column = lr_column_settings->get_column( 'PRICE' ).

  CREATE OBJECT lr_input_field EXPORTING value_fieldname = 'PRICE'.
  lr_column->set_cell_editor( lr_input_field ).

[…]
```

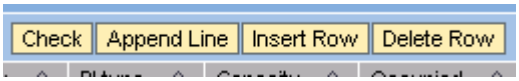
Set table editable.

Additionally the “read only mode” has to be set to abap_false to make the table editable.

```
WDDOINIT()
[…]

* set read only mode to false (and display edit toolbar)
  data: lr_table_settings type ref to if_salv_wd_table_settings.

  lr_table_settings ?= l_value.
  lr_table_settings->set_read_only( abap_false ).

ENDMETHOD.
```

Without this setting the input field will not be editable!

In addition this setting displays the edit toolbar:



Save Changes

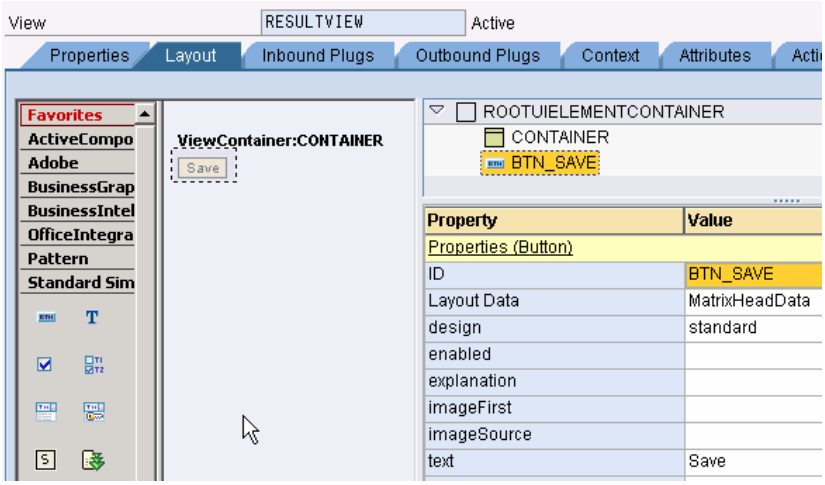
A save button has to be implemented to save the user changes to the database.

Hint: In this tutorial the save functionality is only simulated to not change the flight data model content.

Procedure

Create button SAVE.

Place a save button on view RESULTVIEW with name BTN_SAVE.
Enter “Save” into the text property.



Implement action handler SAVE.

Create action SAVE in the “onAction” property of button BTN_SAVE.

In action handler method ONACTIONSAVE the interface method DATA_CHECK of the ALV component is invoked. If changes have been made then the interface event ON_DATA_CHECK is raised.

ONACTIONSAVE()

```
METHOD onactionsave.

DATA: l_ref_interfacecontroller TYPE REF TO iwci_salv_wd_table .

* Check for changes
l_ref_interfacecontroller = wd_this->wd_cpifc_alv( ).
l_ref_interfacecontroller->data_check(  ).

ENDMETHOD.
```

Implement event handler ONDATACHECK.

Implement an event handler method for the interface event ON_DATA_CHECK with the following functionality:

- Get actual/changed content of the flight table
- Update DB
- Display a success message

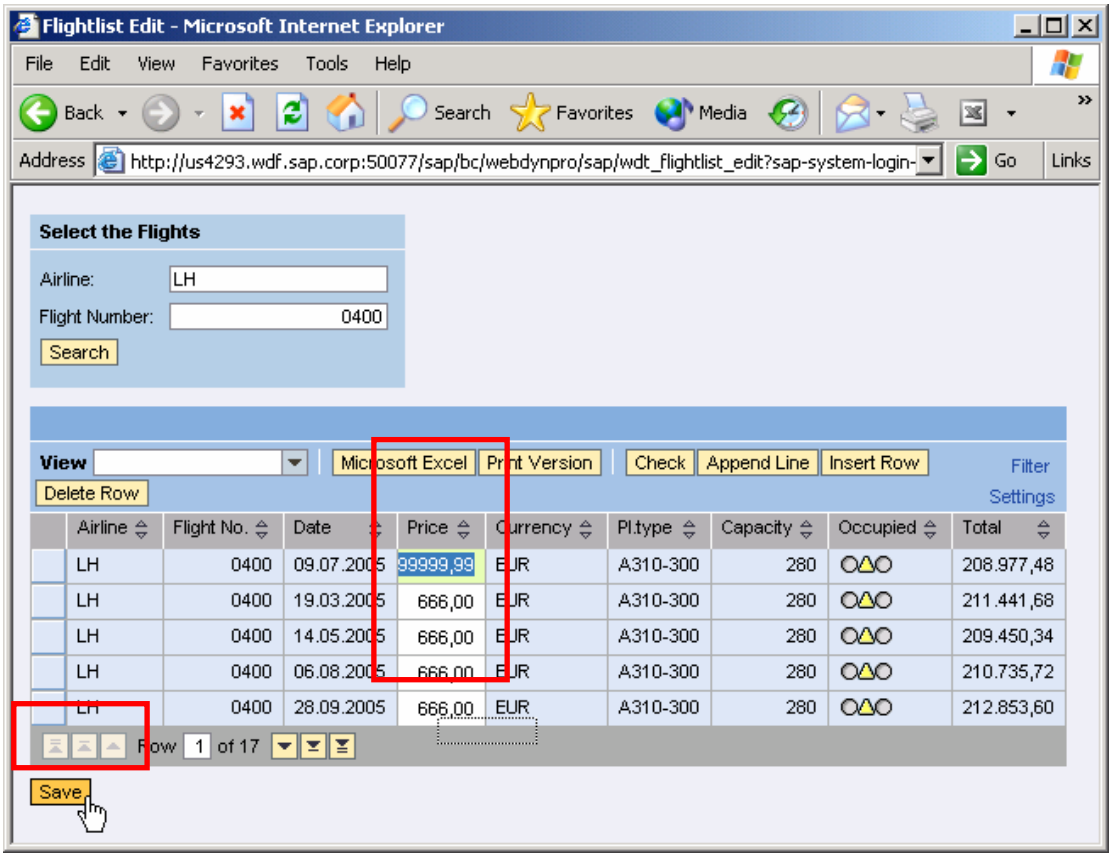
ONDATACHECK()

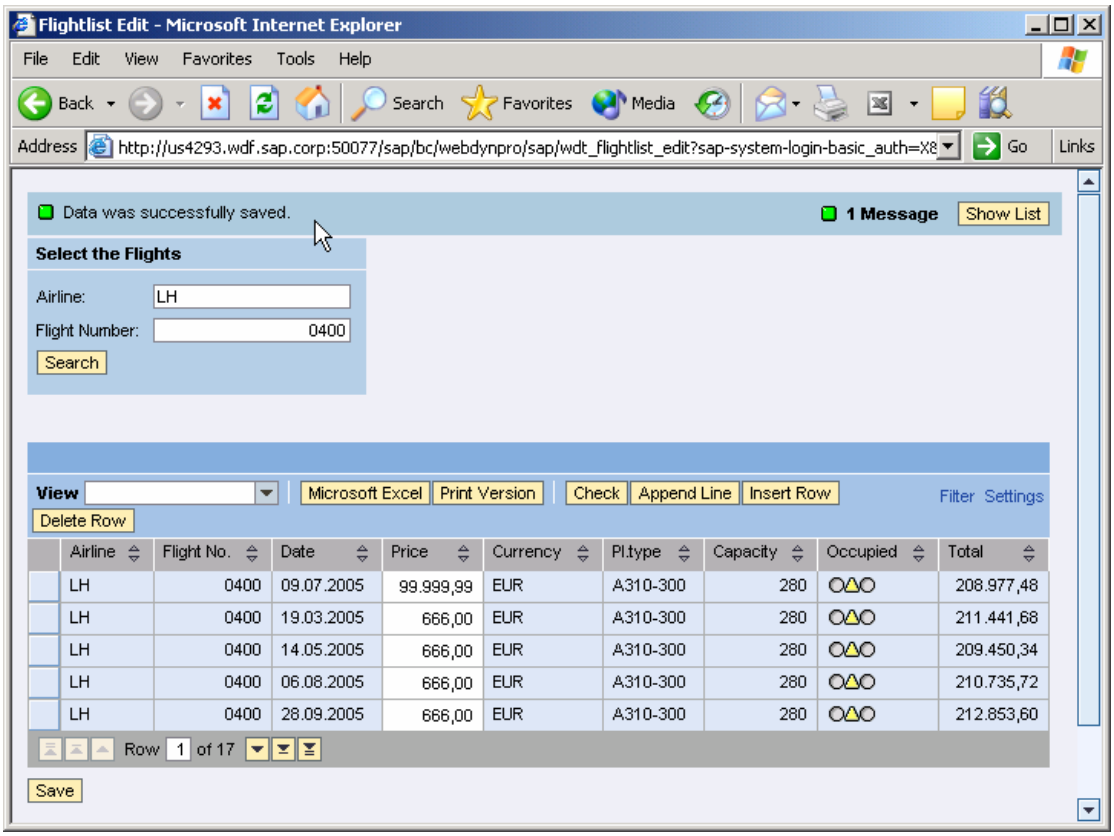
```
METHOD ondatacheck .

  DATA: node_node_flighttab TYPE REF TO if_wd_context_node,
        elem_node_flighttab TYPE REF TO if_wd_context_element,
        lt_sflight TYPE if_resultview=>elements_node_flighttab.

* save data only if no error has occured
  CHECK r_param->t_error_cells IS INITIAL.

* navigate from <CONTEXT> to <NODE_FLIGHTTAB> via lead selection
  node_node_flighttab
    = wd_context->get_child_node( name = `NODE_FLIGHTTAB` ).

* get data from context node <NODE_FLIGHTTAB>
  node_node_flighttab->get_static_attributes_table(
                          IMPORTING table = lt_sflight ).

* save data to database
* update... => only simlate, to not change the flight data model
* content!

* Create success message
* get message manager
  DATA: l_current_controller TYPE REF TO if_wd_controller,
        l_message_manager    TYPE REF TO if_wd_message_manager.

  l_current_controller ?= wd_this->wd_get_api( ).
```

```
  CALL METHOD l_current_controller->get_message_manager
    RECEIVING
      message_manager = l_message_manager.

* report message
  CALL METHOD l_message_manager->report_success
    EXPORTING
      message_text = 'Data was successfully saved.'.

ENDMETHOD.
```

Test Your Web Dynpro Application

The result will look like the following:





Author Bio

Claudia Dangers is a senior development consultant in SAP's Software Technology and Development department. Since she joined SAP in 1999 she has worked on numerous projects and gained practical experience in ABAP and BSP development, in the creation of concepts, in coaching and code reviews, and as a sub-project lead and training instructor. Claudia is very interested in new technologies. Currently she is dealing with Web Dynpro ABAP, kernel-based BADI's and the Switch and Enhancement Framework.