Messages internationalization with Web Dynpro ABAP

Applies to:

Web Dynpro ABAP. For more information, visit the Web Dynpro ABAP homepage.

Summary

This tutorial show the way we can use Assistance Class and Message Class to create Multilanguage massages and the use of UI Elements: PhaseIndicator, LinkToAction, ViewConteinerUIElement, MessageArea, DropDownByKey, InputField, Lebel, Group, HorizontalGutter, TransparentConteiner and ABAP Dictionary Repository Objects: Domains and Data elements

Author: Drd. Ing. Cristea Ana Daniela

Company: Drd. Of University Politehnica Timisoara, Faculty of Automation and Computer Science and Engineering, Romania

Created on: 3 February 2009

Author Bio

Four years Assistant to University of Timisoara, faculty of Engineering of Hunedoara, author and coauthor of 3 Books: Computer Programming, Computer utilization, Interface and peripheral. From October 2006 drd. Of University “Politehnica” Timisoara where I make my study in ABAP POO, Web Dynpro ABAP and SAP NetWeaver Development Studio respective Web Dynpro Java.

Table of Contents

Objective Exercise

The application shows a (highly simplified) online car booking form. It allows users to enter their name, the date they want to travel, their departure city, and preferred car type.

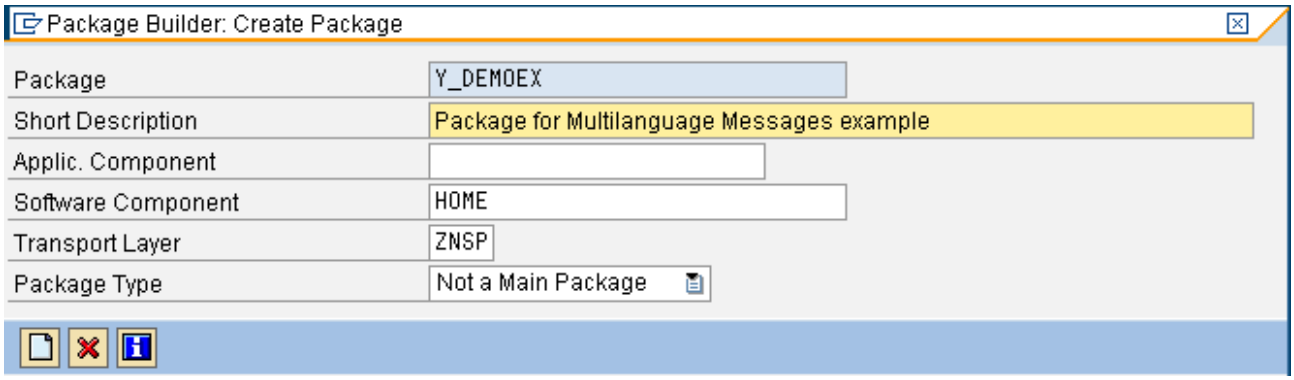
We create an application that use Message Class and Assistant Class. We create four views, a main view and 3 secondary views where we build the user interface with user entry validation. When the inputs are not proper we have to show a Multilanguage message. For the User Interface we use the UI Elements: DropDownByKey, InputField, Lebel, Group, HorizontalGutter, TransparentConteiner, PhaseIndicator, LinkToAction, MessageArea and ABAP Dictionary Repository Objects: Domains and Data elements.

Prerequisites

Basic knowledge of programming in ABAP and Web Dynpro for ABAP is required.

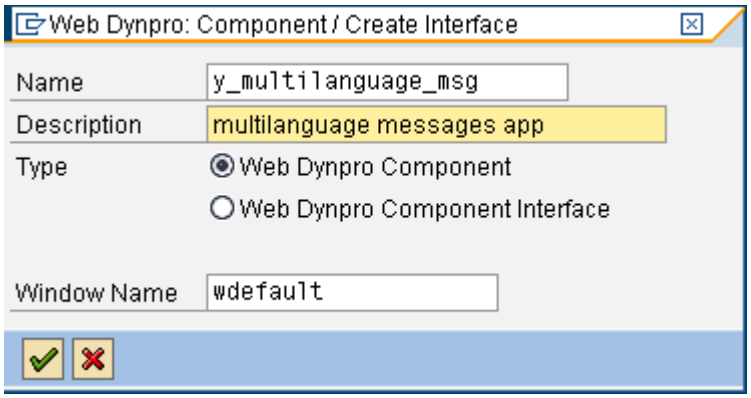
Create a Package

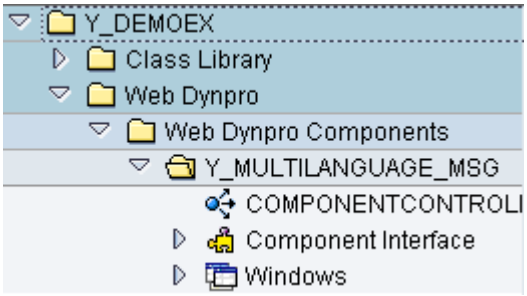
We can use the predefined package $TMP where we can make local test applications or we can create our separate package to store all Repository Objects that we have to create for this application. In this case we have chosen to create a separate package named Y_DEMOX of type “Not a Main Package”. We have chose this type because we want to store Repository Objects and not others Packages.



Create Web Dynpro component

First we will create a new Web Dynpro component in SE80 with the name y_multilanguage_msg.



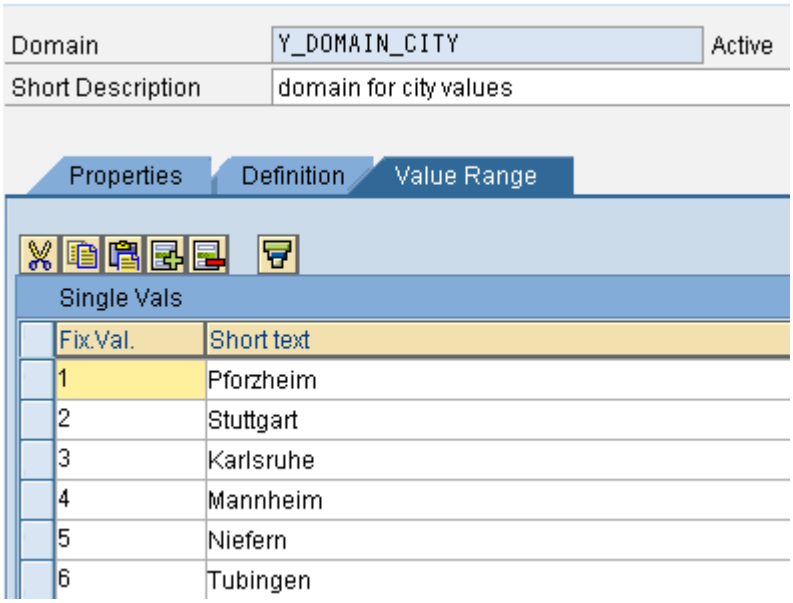


After that we create the context nodes and context attributes in Component controller

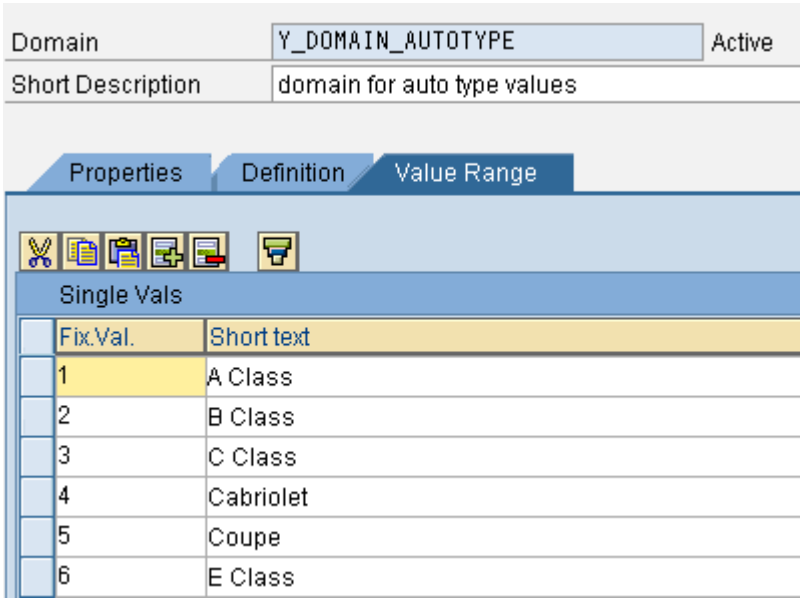
Create context attributes in component controller

Create two nodes in the context of the component controller. Named DATE and DATARESULT. Node DATE has four attributes named NAME, DATE, CITY, AUTOTYPE. The attribute NAME it is of type string, the attribute DATE it is of type datum. The attributes CITY and AUTOTYPE are data elements specific to our application because we want to use Domains for CITY and AUTOTYPE. A Domain it is a repository object in the ABAP Dictionary. We can create a domain with the transaction SE11 (ABAP Dictionary Maintenance) or we can create with right click of the package name and after that we chose from the list create->dictionary object->domain. We create a Domain for the CITY and a Domain for the AUTOTYPE.

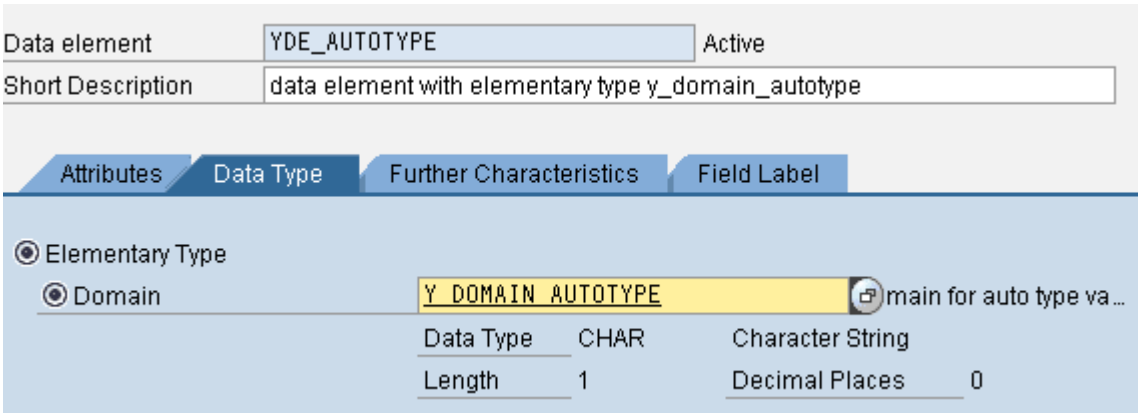
First Domain for CITY data type CHAR, No. Characters 1:

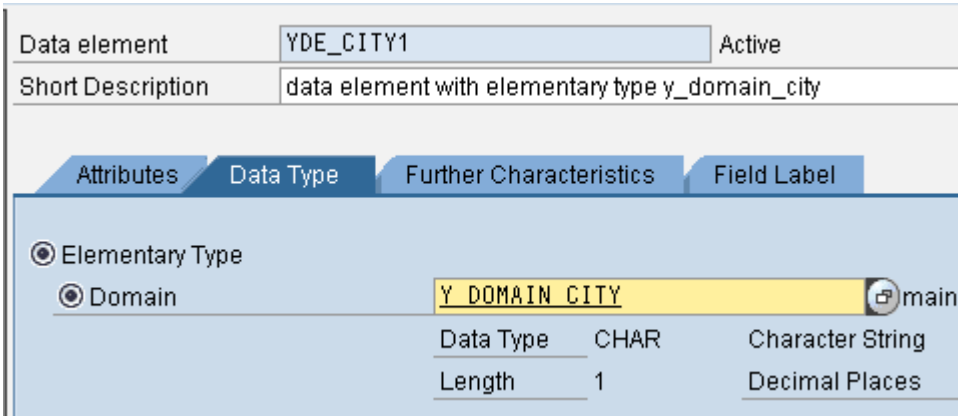


Second Domain for AUTOTYPE data type CHAR No Characters 1:



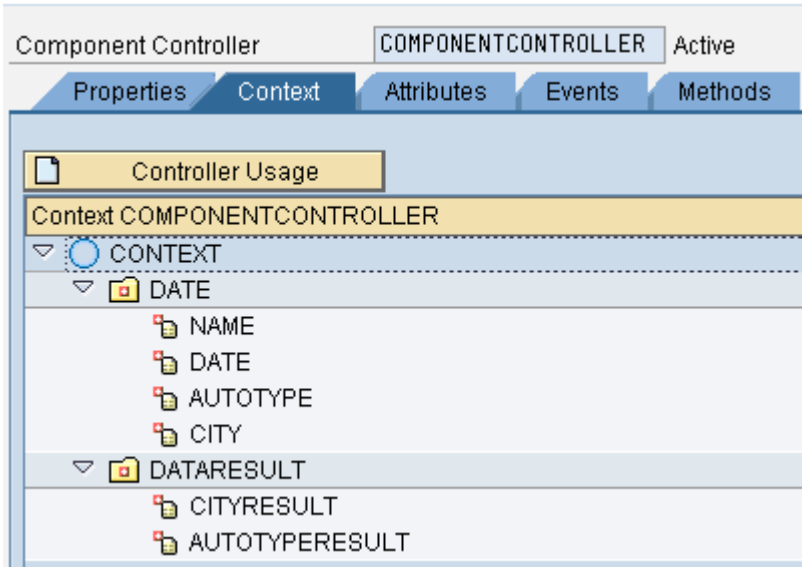
To use these Domains in our Web Dynpro application we have to define two data elements of elementary type Domain:





The secondary node is DATARESULT with two attributes of type string: CITYRESULT and AUTOTYPERESULT.

The context in Component controller has the structure:



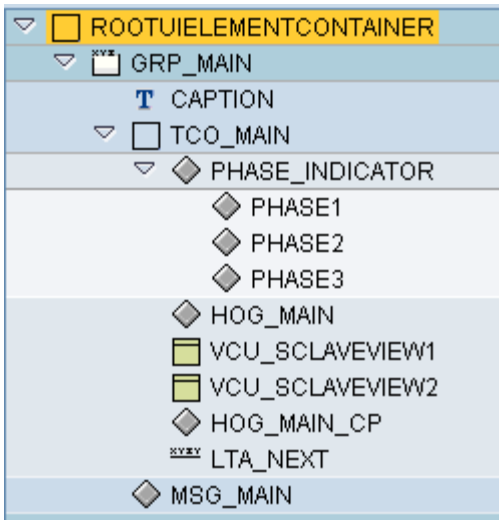
Create the User Interface

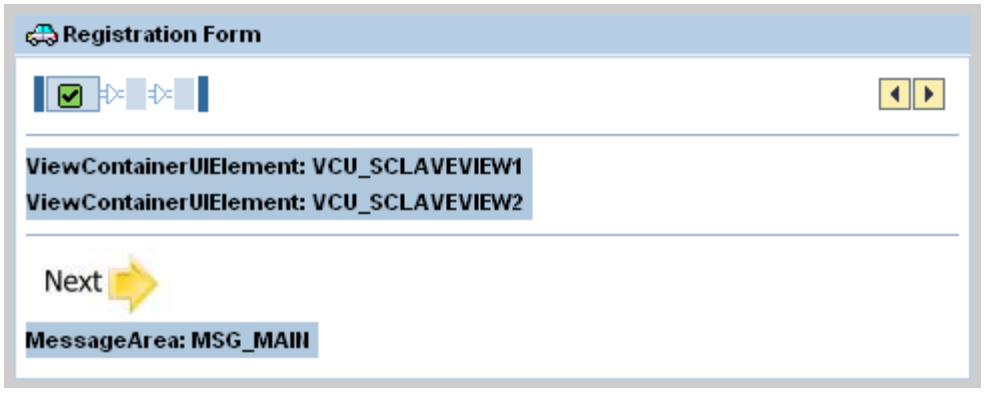
We need to create views for our application. The main view will contain the PhaseIndicator UI element with three phases and the others views will contain the proper UI Interface need for every phase of the PhaseIndicator UI Element.

Create Main View

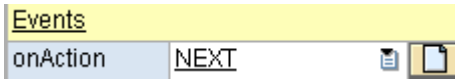
Create a main view named MAINVIEW. This view will contain a PhaseIndicator UI Element with three Phases, a LinkToAction for navigation, two containers ViewConteinerUIElement to integrate the other views, a Group UI Element where we integrate all the others UI Elements and a MessageArea UI Element where the proper messages is shown.

Main View Layout





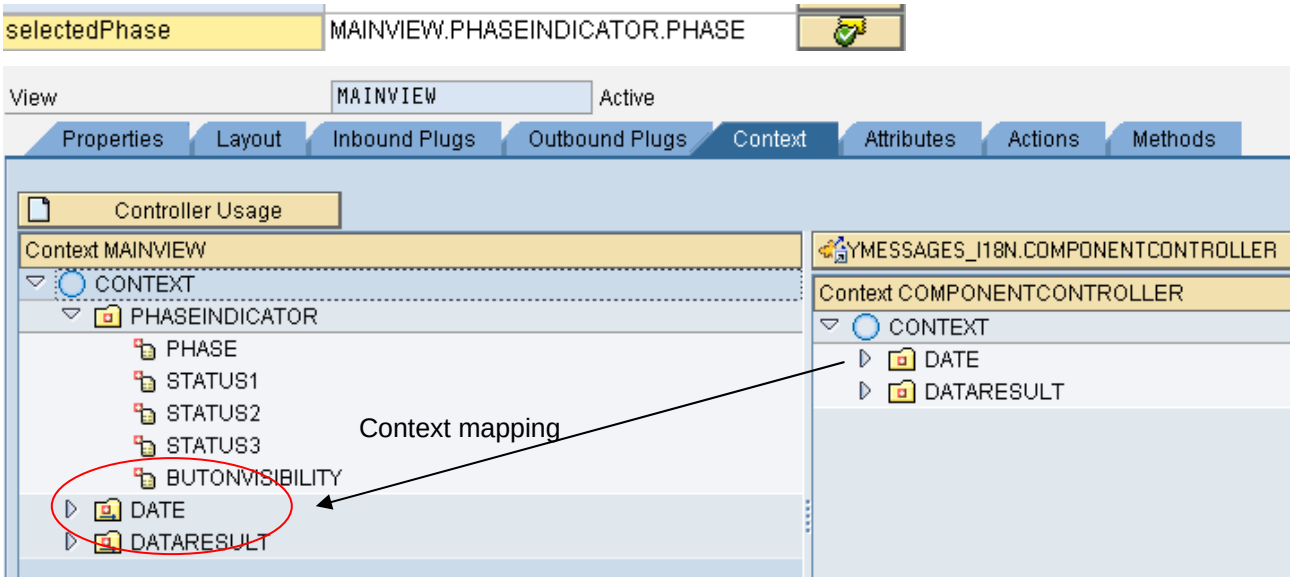
The Button Next it is a LinkToAction UI Element with a proper image and with the action



The UI Element PhaseIndicator (More details - reference 1), in our case with three Phases, is used to displays the steps in a wizard.

Main View Context

After the Layout section we create the context section. We make a context mapping from the component controller and we define a new context node named PHASEINDICATOR with five attributes. We create this node in Context View, because we need only in this View to set dynamic the property of the PhaseIndicator UI element and to set the property visibility of the LinkToAction UI Element. The Phase attribute has the type string and we use to set dynamic the property selected Phase of the PhaseIndicator UI Element:



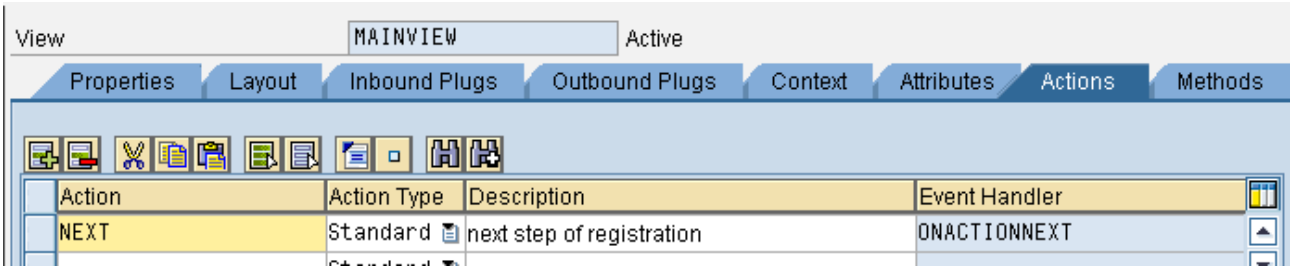
The attributes STATUS1, STATUS2, STATUS3 are use to set the properties Status of the Phases and have the type:

Type	WDUI_PHASE_STATUS

BUTONVISIBILITY UI Element it is use to set the property Visibility of the LinkToAction UI Element and have the type:

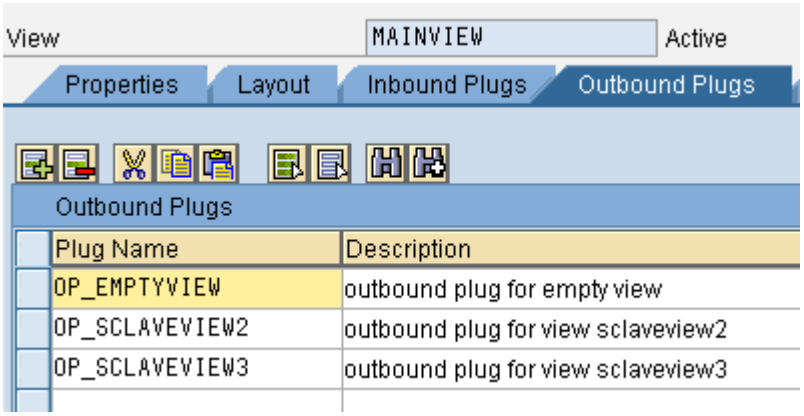
Type	WDUI_VISIBILITY

Main View Actions

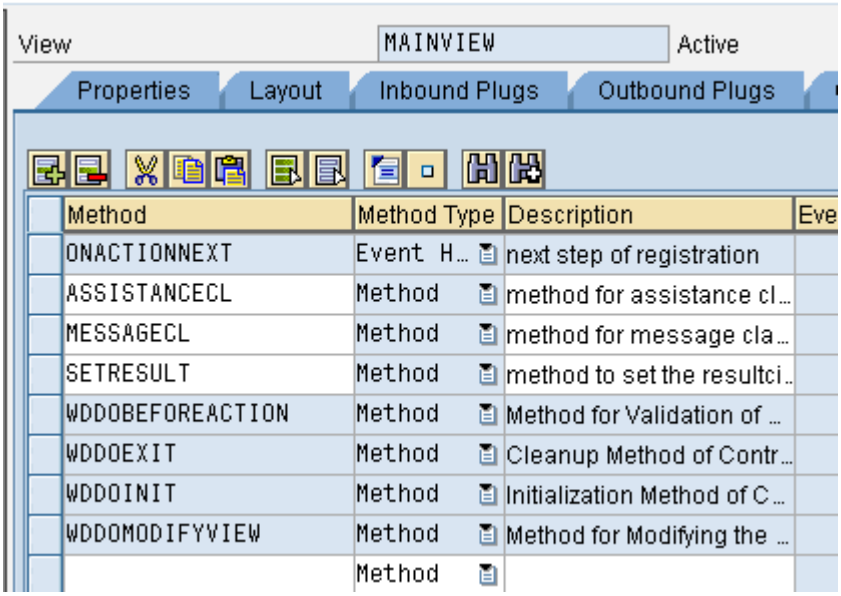


Next it is the action for the LinkToAction UI ELement

Main View Outboud Plugs

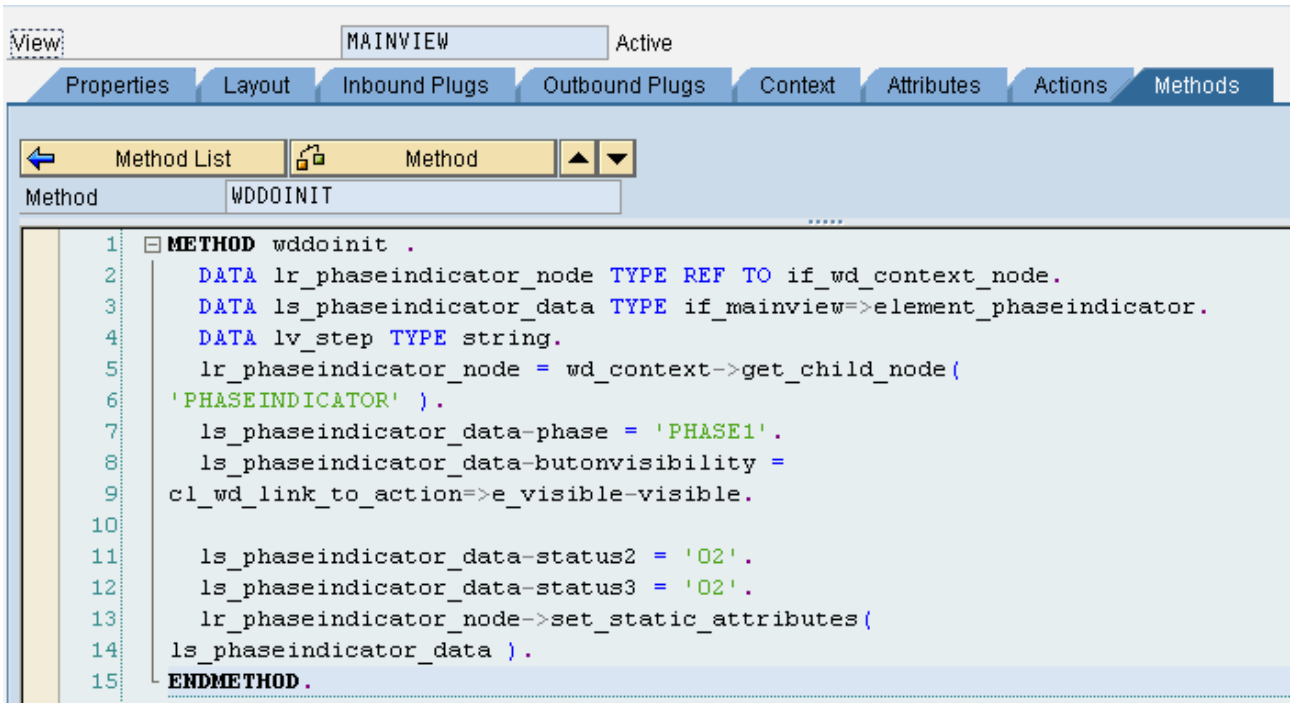


Main View Methods



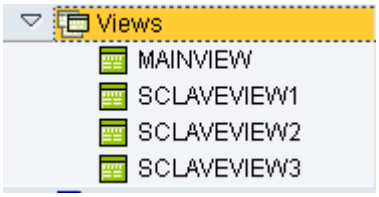
We implement the methods onactionnext, assistancecl, messagecl and setresult after the user interface.

The Methods: wddobeforeaction, wddoexit, wddoinit, wddomodifyview are the default methods, generated from framework so named Hook Methods. We use wddoinit( ) Hook Method of the View Controller to initializing our attributes:



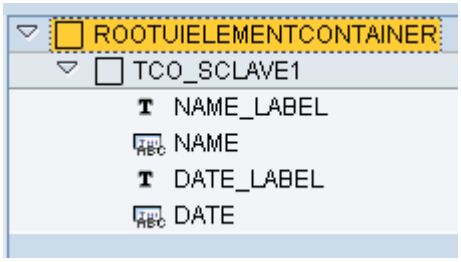
Create others views

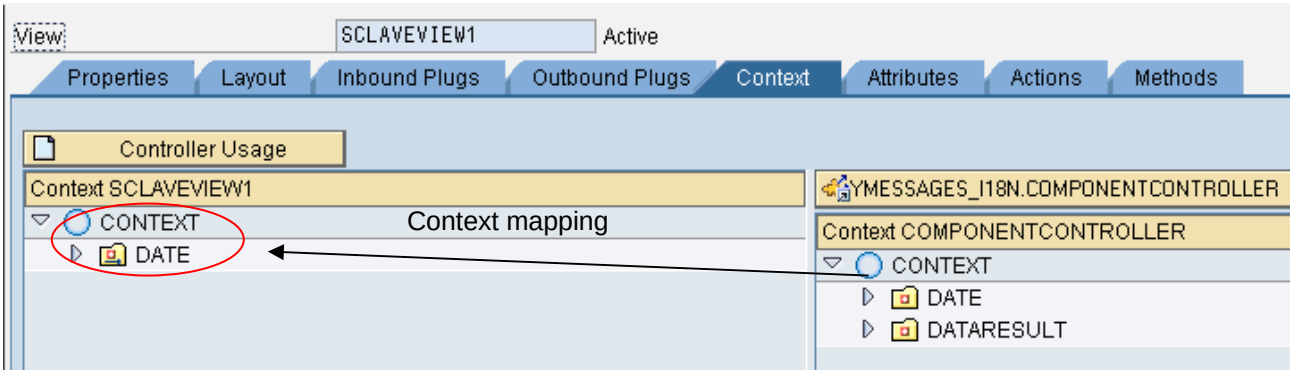
We create tree Views that we integrate in the ViewConteinerUIELements definite in MAINVIEW.



Create secondary View SCLAVEVIEW1

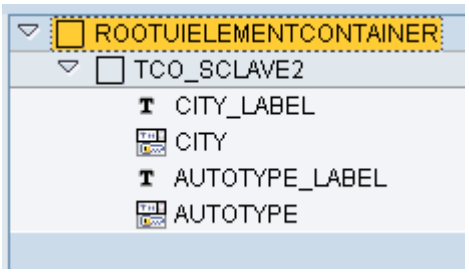
SCLAVEVIEW1 have the Layout and Context:

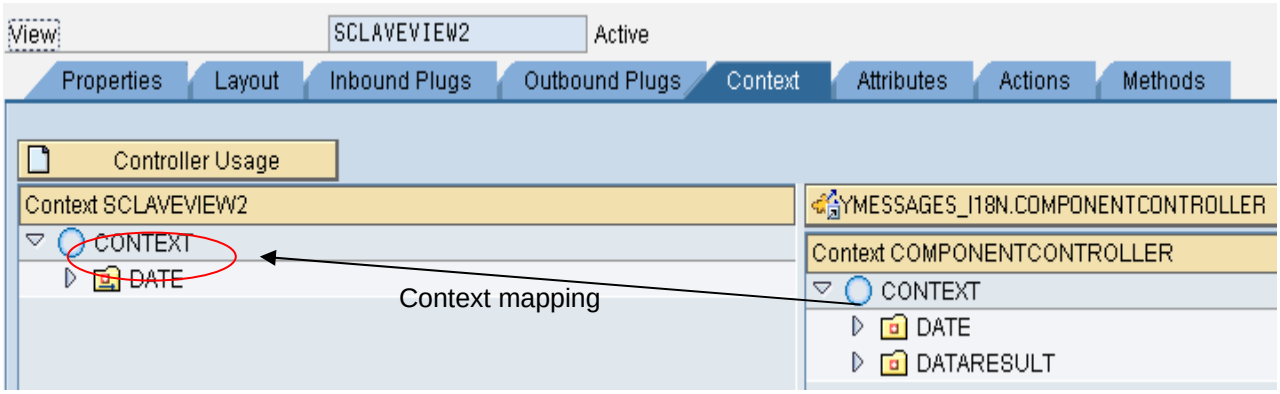


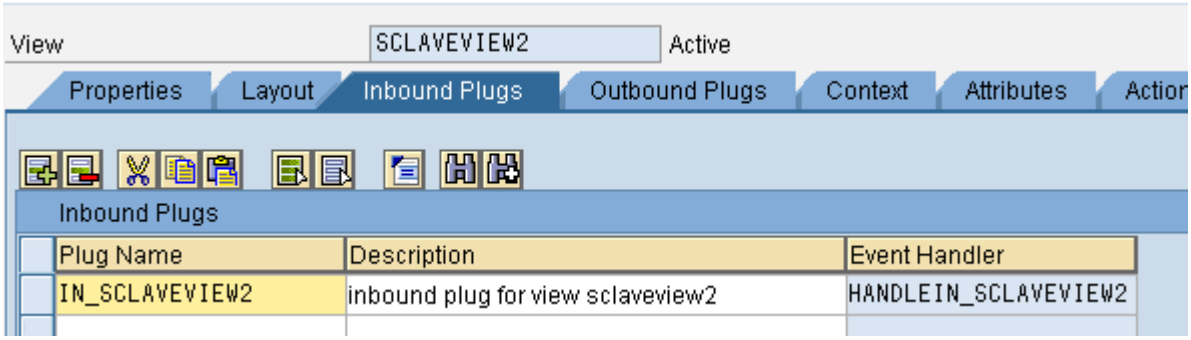


Create secondary View SCLAVEVIEW2

SCLAVEVIEW2 have the Layout, Context and Inbound Plugs:

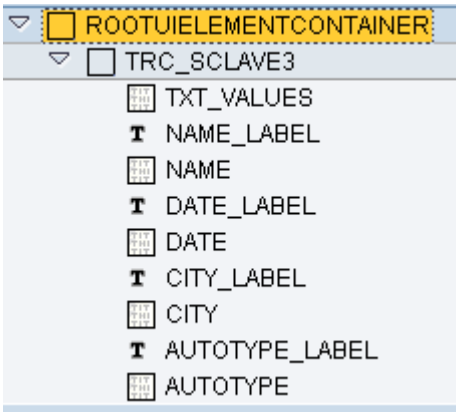


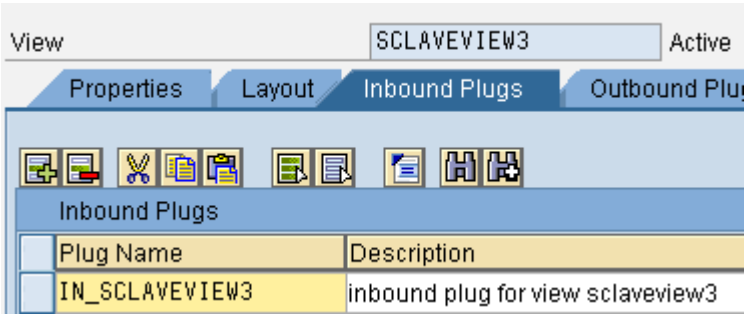


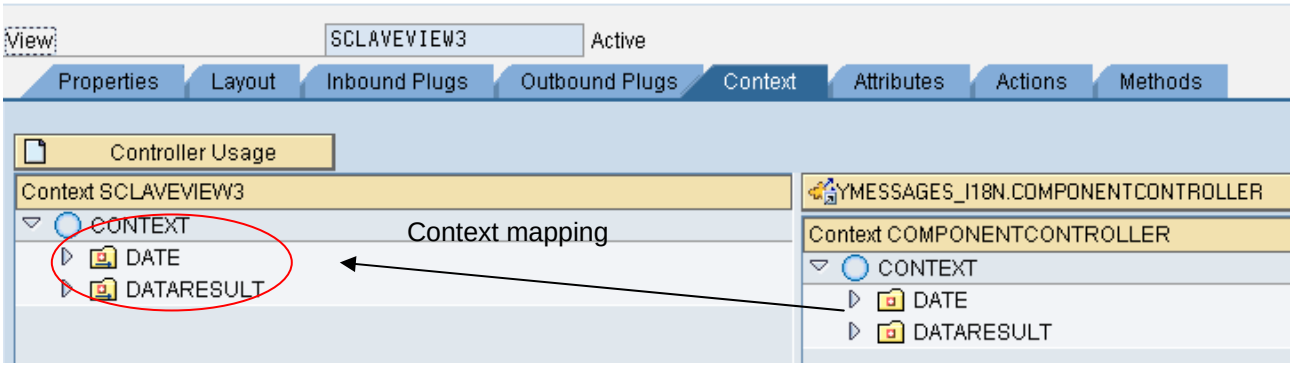

View
SCLAVEVIEW2
Active
Properties
Layout
Inbound Plugs
Outbound Plugs
Context
Attributes
Actior
Inbound Plugs
Plug Name
Description
Event Handler
IN_SCLAVEVIEW2
inbound plug for view sclaveview2
HANDLEIN_SCLAVEVIEW2

Create secondary View SCLAVEVIEW3

SCLAVEVIEW3 have the Layout, Context and Inbound Plugs:

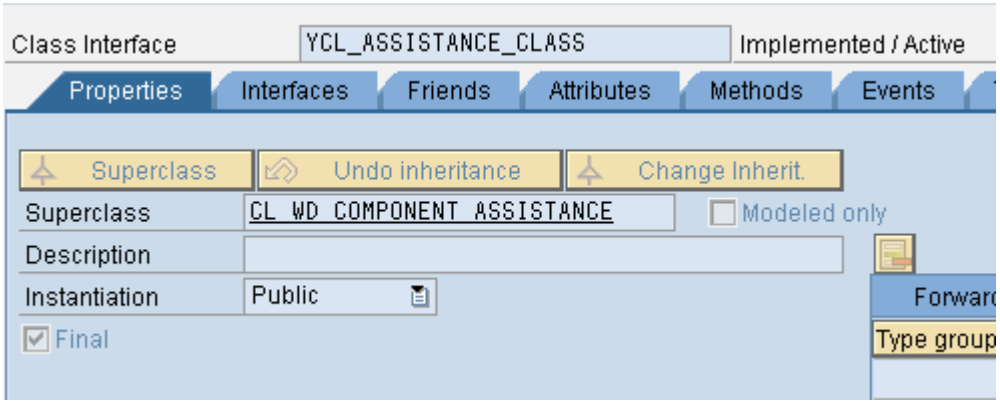






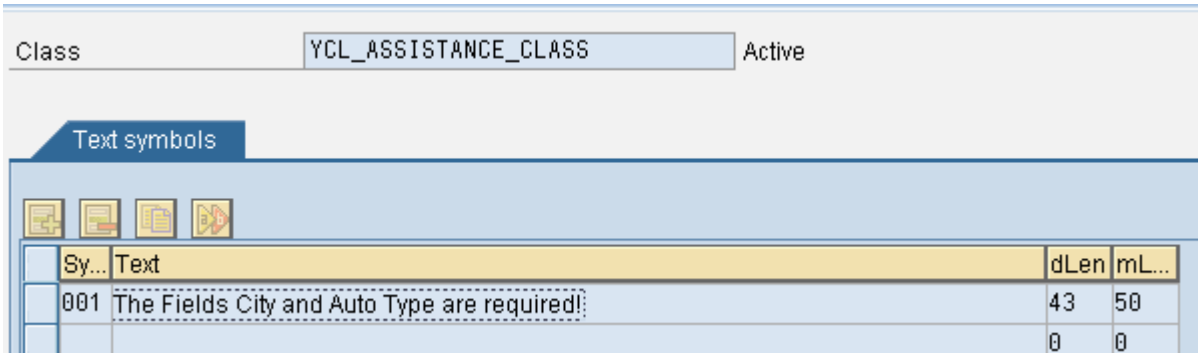
Create the Assistance Class and assign to Web Dynpro Component

We create a class YCL_ASSISTANCE_CLASS that inheriting the super class CL_WD_COMPONENT_ASSISTANCE

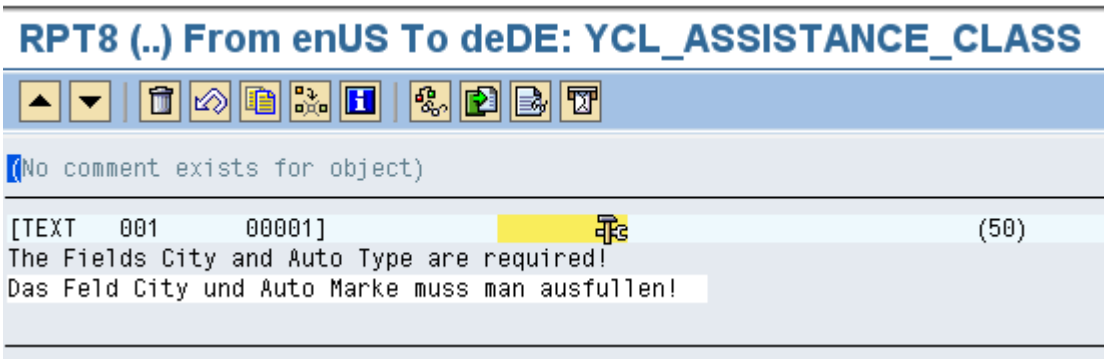


We use this class to work with text symbols. More detail – reference 2

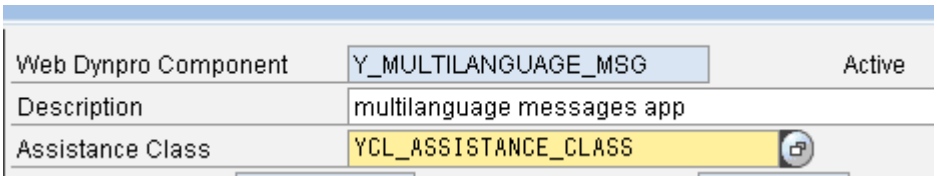
We define a text symbol:



Because we want to create Multilanguage messages we have to translate this text symbol in German:

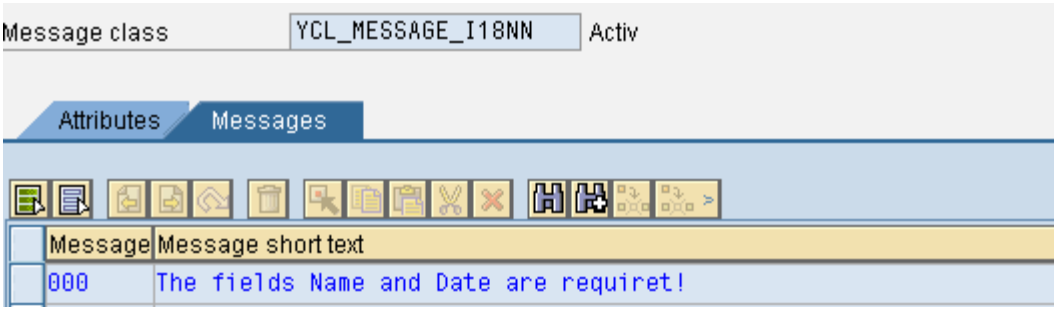


After creation we can assign to our Web Dynpro Component:

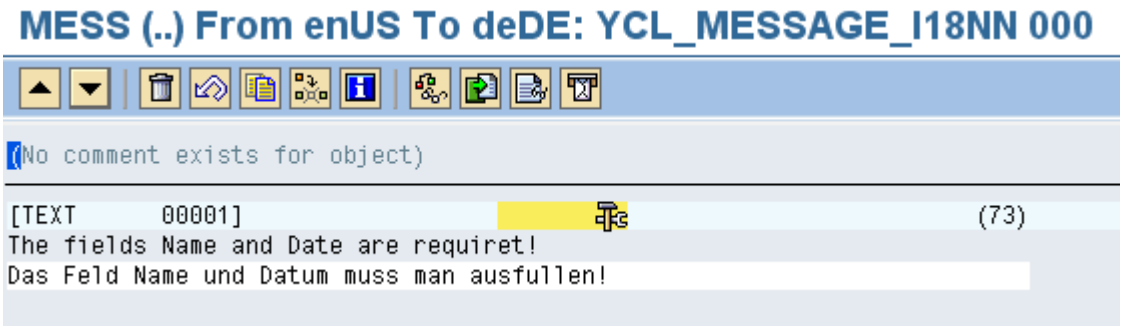


Create the Message Class

We create a message class YCL_MESSAGE_I18N. More detail – reference 3



We translate this message in German



Implementation of the Methods from MAINVIEW view

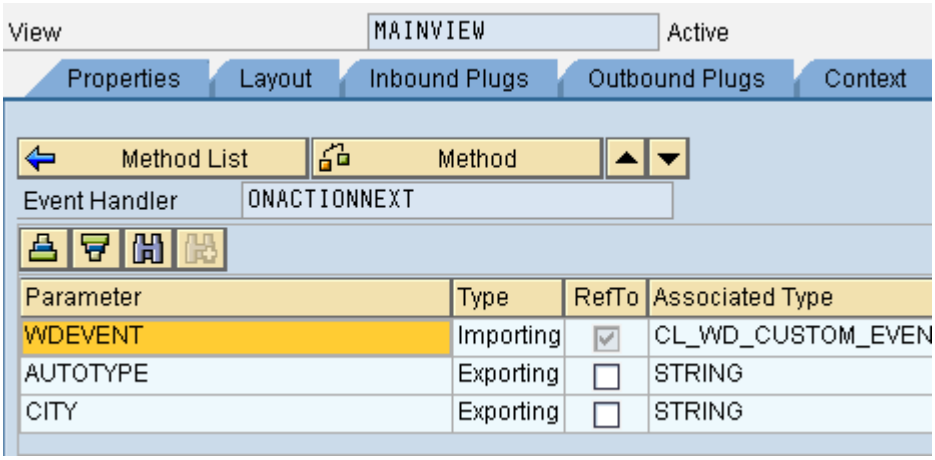
We have to implement the methods:

ONACTIONNEXT	Event H…	next step of registration
ASSISTANCECL	Method	method for assistance cl…
MESSAGECL	Method	method for message cla…
SETRESULT	Method	method to set the resultci…

from MAIVIEW.

Onactionnext Event Handler

The action Next it is assigned to the onAction property of the LinkToAction UI Element.




```
METHOD onactionnext .
  DATA lr_phaseindicator_node TYPE REF TO if_wd_context_node.
  DATA ls_phaseindicator_data TYPE if_mainview=>element_phaseindicator.
  DATA lv_phase TYPE string.
  DATA ls_compcontroller_data TYPE
if_mainview=>element_date.
  DATA lr_compcontroller_node TYPE REF TO if_wd_context_node.
  DATA lv_name TYPE string.
  DATA lv_date TYPE datum.
  DATA lv_city TYPE string.
  DATA lv_autotype TYPE string.
  lr_compcontroller_node = wd_context->get_child_node(
 'DATE' ).
  lr_compcontroller_node->get_attribute( EXPORTING name = 'NAME'
    IMPORTING value = lv_name  ).
  lr_compcontroller_node->get_attribute( EXPORTING name = 'DATE'
```

```
    IMPORTING value = lv_date  ).
    lr_compcontroller_node->get_attribute( EXPORTING name = 'CITY'
       IMPORTING value = lv_city  ).
    lr_compcontroller_node->get_attribute( EXPORTING name = 'AUTOTYPE'
      IMPORTING value = lv_autotype  ).
    lr_phaseindicator_node = wd_context->get_child_node(
'PHASEINDICATOR' ).
    lr_phaseindicator_node->get_attribute( EXPORTING name = 'PHASE'
      IMPORTING value = lv_phase  ).
    CASE lv_phase.
      WHEN 'PHASE1'.
        IF lv_name IS INITIAL OR lv_date IS INITIAL.
          wd_this->messagecl( ).
          EXIT.
        ELSE.
          ls_phaseindicator_data-phase = 'PHASE2'.
          ls_phaseindicator_data-status2 = '01'.
          ls_phaseindicator_data-status3 = '02'.
          ls_phaseindicator_data-butonvisibility =
    cl_wd_link_to_action=>e_visible-visible.
          wd_this->fire_op_sclaveview2_plg( ).
        ENDIF.
      WHEN  'PHASE2'.
        IF lv_city IS INITIAL OR lv_autotype IS INITIAL.
          wd_this->assistancecl( ).
          EXIT.
        ELSE.
          ls_phaseindicator_data-phase = 'PHASE3'.
          ls_phaseindicator_data-status2 = '01'.
          ls_phaseindicator_data-status3 = '01'.
          wd_this->setresult(  city = lv_city
                                      autotype = lv_autotype ).
          wd_this->fire_op_sclaveview3_plg( ).
          wd_this->fire_op_emptyview_plg( ).
        ENDIF.
      WHEN OTHERS.
        ls_phaseindicator_data-butonvisibility =
  cl_wd_link_to_action=>e_visible-none.
    ENDCASE.
    lr_phaseindicator_node->set_static_attributes(
  ls_phaseindicator_data ).
  ENDMETHOD.
```

We have to verify if the user fill all the entries. If a field it is empty we have to show a proper message. We have created a message class and an assistance class. We want to build an application with Multilanguage messages.

If the fields name or data are empty we use message class YCL_MESSAGE_I18N to show the message 000.

```
    IF lv_name IS INITIAL OR lv_date IS INITIAL.
      wd_this->messagecl( ).
```

For that we have implement the user-defined method without parameters messagecl( )

This instance methods are called in the own controller with the self-reference wd_this.

If the fields city or auto type are empty we use assistance class YCL_ASSISTANCE_CLASS to show the text symbol definite here.

```
IF lv_city IS INITIAL OR lv_autotype IS INITIAL.
  wd_this->assistancecl( ).
```

For that we have implement the user-defined method without parameters `assistancecl()`

If all the entries are filled we go to the end point. To show the data chose from Drop Dawn list we have implement the user-defined method setresult( ) with two parameters.

```
wd_this->setresult(  city = lv_city
                             autotype = lv_autotype ).
```

User-defined assistancecl method

```
METHOD assistancecl .
  DATA: l_current_controller TYPE REF TO if_wd_controller.
  DATA   l_message_manager    TYPE REF TO if_wd_message_manager.
  DATA text TYPE string.
  text =  wd_assist->if_wd_component_assistance~get_text( key = '001' ).
  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_error_message
    EXPORTING
      message_text             = text
*     PARAMS                   =
*     MSG_USER_DATA            =
*     VIEW                     =
*     SHOW_AS_POPUP            =
*     IS_PERMANENT             =
*     SCOPE_PERMANENT_MSG      =
*     CONTROLLER_PERMANENT_MSG =
      .
ENDMETHOD.
```

We can use the Web Dynpro Code Wizard  to help us for this code generation.

User-defined messagecl method

```
METHOD messagecl .
  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_t100_message
    EXPORTING
      msgid                     = 'YCL_MESSAGE_I18NN'
      msgno                     = '000'
      msgty                     = 'E'
*                 P1                          =
*                 P2                          =
*                 P3                          =
*                 P4                          =
*                 MSG_USER_DATA               =
*                 VIEW                        =
*                 SHOW_AS_POPUP               =
*                 IS_PERMANENT                =
*                 SCOPE_PERMANENT_MSG         =
*                 CONTROLLER_PERMANENT_MSG    =
.
ENDMETHOD.
```

We can use the Web Dynpro Code Wizard  to help us for this code generation.

User-defined setresult method

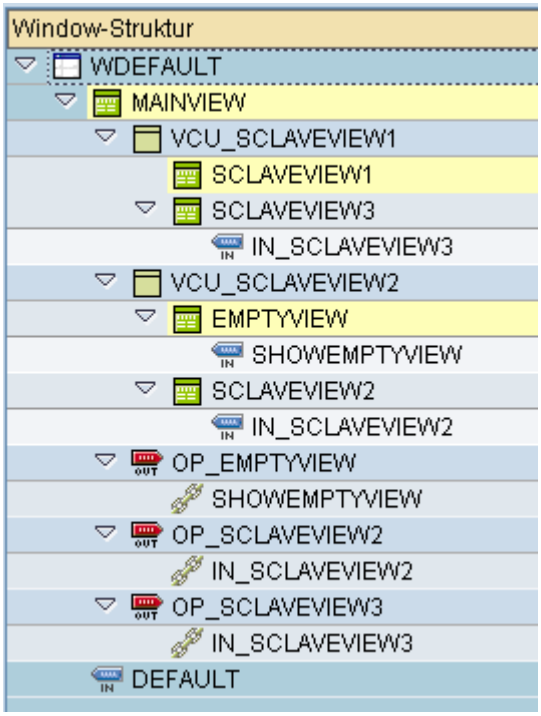
This method has two parameters: city and autotype:

```
METHOD setresult .
  DATA lr_node TYPE REF TO if_wd_context_node.
  DATA ls_data TYPE if_mainview=>element_dataresult.
  lr_node = wd_context->get_child_node( 'DATARESULT' ).
  CASE city.
    WHEN '1'.
      ls_data-cityresult = 'Pforzheim'.
    WHEN '2'.
      ls_data-cityresult = 'Stuttgart'.
    WHEN '3'.
      ls_data-cityresult = 'Karlsruhe'.
    WHEN '4'.
      ls_data-cityresult = 'Mannheim'.
    WHEN '5'.
      ls_data-cityresult = 'Niefern'.
    WHEN OTHERS.
      ls_data-cityresult = 'Tubingen'.
  ENDCASE.
  CASE autotype.
    WHEN '1'.
      ls_data-autotyperesult = 'A Class'.
```

```
        WHEN '2'.
          ls_data-autotyperesult = 'B Class'.
        WHEN '3'.
          ls_data-autotyperesult = 'C Class'.
        WHEN '4'.
          ls_data-autotyperesult = 'Cabriolet'.
        WHEN '5'.
          ls_data-autotyperesult = 'Coupe'.
        WHEN OTHERS.
          ls_data-autotyperesult = 'E Class'.
      ENDCASE.
      lr_node->set_static_attributes( ls_data ).
    ENDMETHOD.
```

Embed views in window

The next step is to embed the main view in the WDDEFAULT window of the component. After that we have to embedded the proper views in the ViewConteinerUIElements



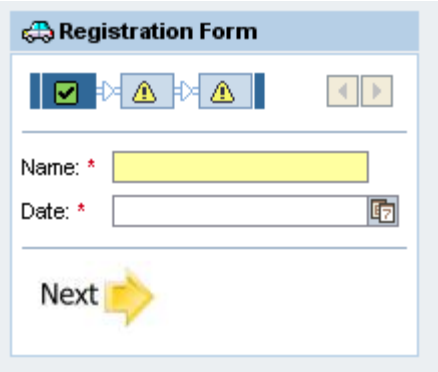
We have two UI Elements ViewConteinerUIElements and we use Emptyview when we want to show just a view at a moment.

Result

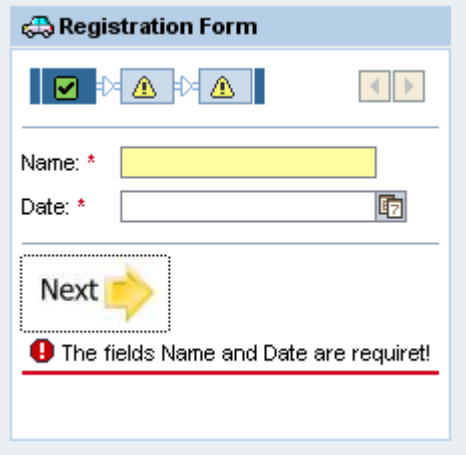
After we create a Web Dynpro application we can execute the component

English result

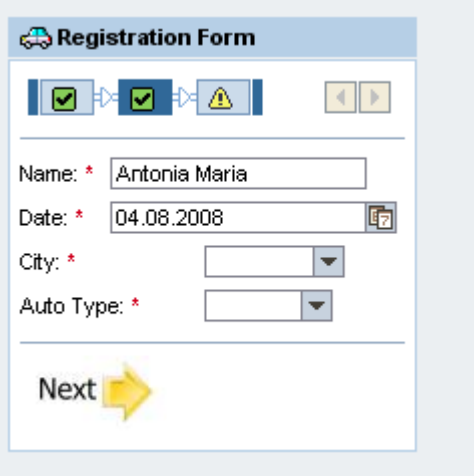
Step1



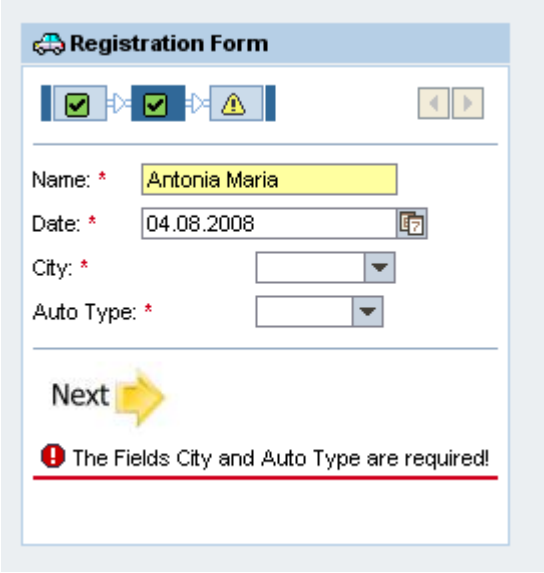
When we push the Button Next a message it is show – Message Class:



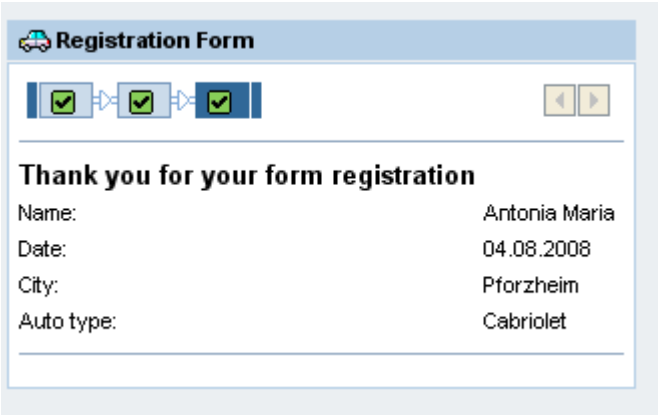
After we enter the proper values we go to the next step:



If we don't fill the fields a message it is show – Assistance Class:



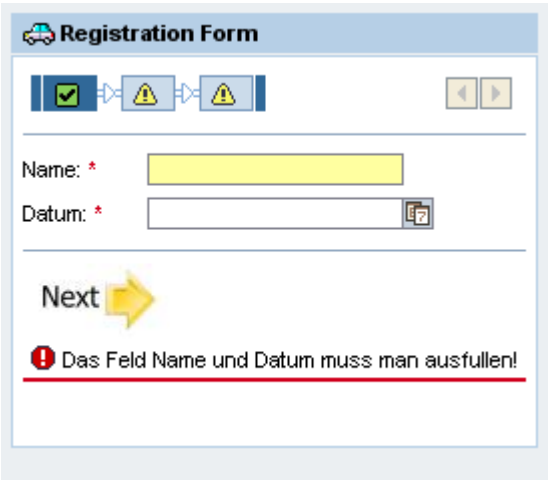
If we chose the proper values we go to the last step:



German result

If we change the logon language in German we obtain the messages in German without need to recoding.

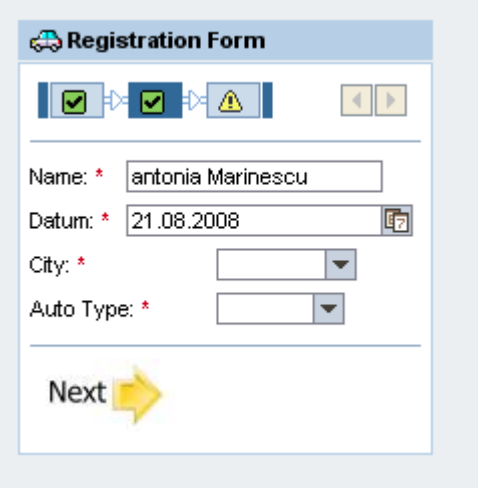
We have made the proper translation when we have create the Message Class and the Assistance Class.



We have create Multilanguage just the messages. The others labels, capitation are not Multilanguage but we can easy create Multilanguage too. In this tutorial we want to show the way we can build Multilanguage messages.

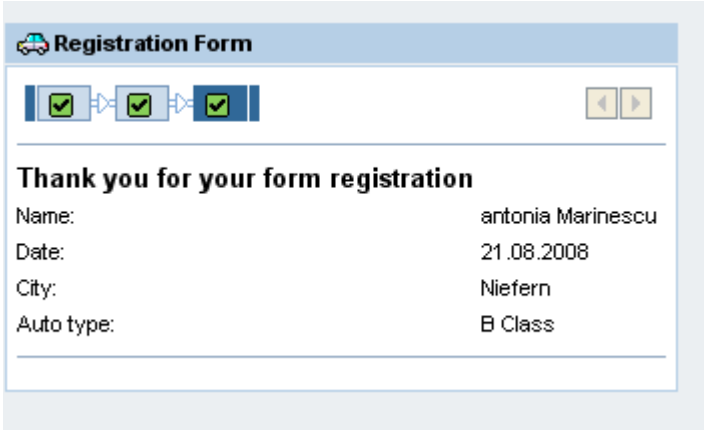
The label Date it is automatic translate because it is a label for an InputField bind to an attribute of type Datum.

After we enter the proper values we go to the next step



If we don’t fill the fields a message is shown – Assistance Class:

If we choose the proper values we go to the last step:


Registration Form
Thank you for your form registration
Name:
antonia Marinescu
Date:
21.08.2008
City:
Niefern
Auto type:
B Class

Related Content

- http://help.sap.com/saphelp_nw04s/helpdata/en/03/ac884118aa1709e10000000a155106/frameset.htm
- http://help.sap.com/saphelp_nw04s/helpdata/en/1e/401ad6ee3c11d1951d0000e8353423/frameset.htm
- https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/63a47dd9-0b01-0010-3d8e-de27242bf011
- 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.