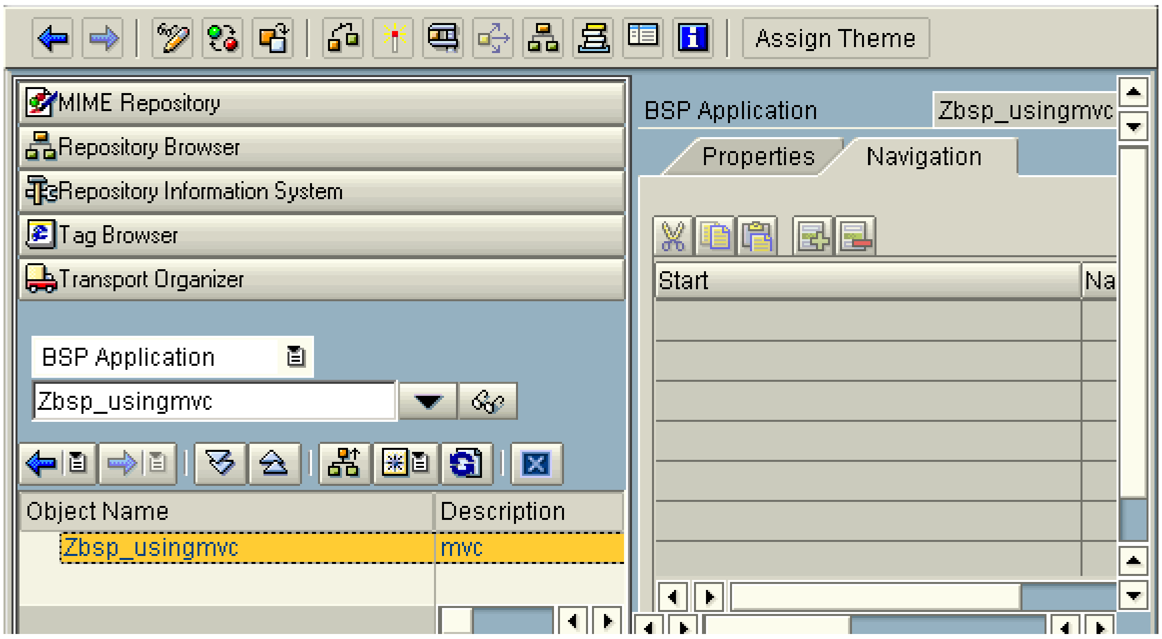
Creating a BSP using the Model View Controller (MVC) technique

Tutorial 1 - Creating the View & the controller (htm page & events)

Step 1 - Create new BSP Application

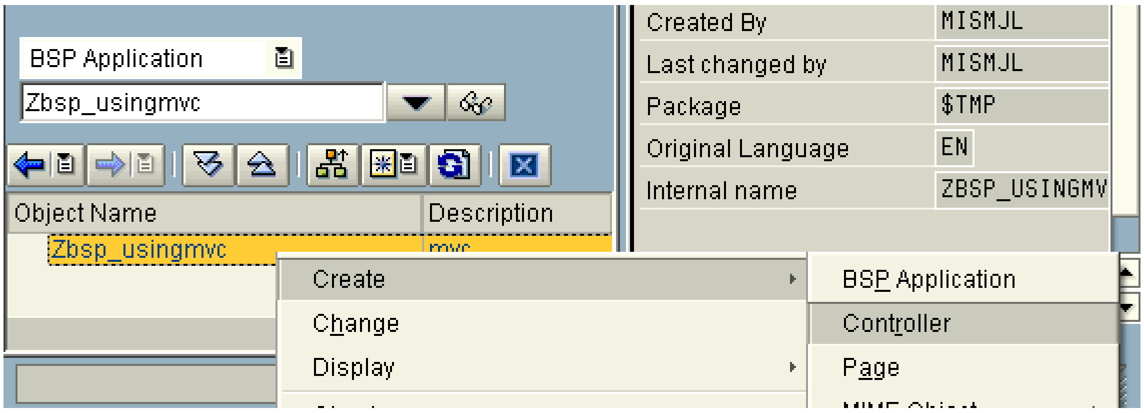
Using SE80 create BSP Application (I.e. Zbsp_usingmvc).



Step 2 - Create new Controller (main.do)

Right click on BSP application name and select create->controller. Enter name main.do or your own

name + description. Press the green tick to continue

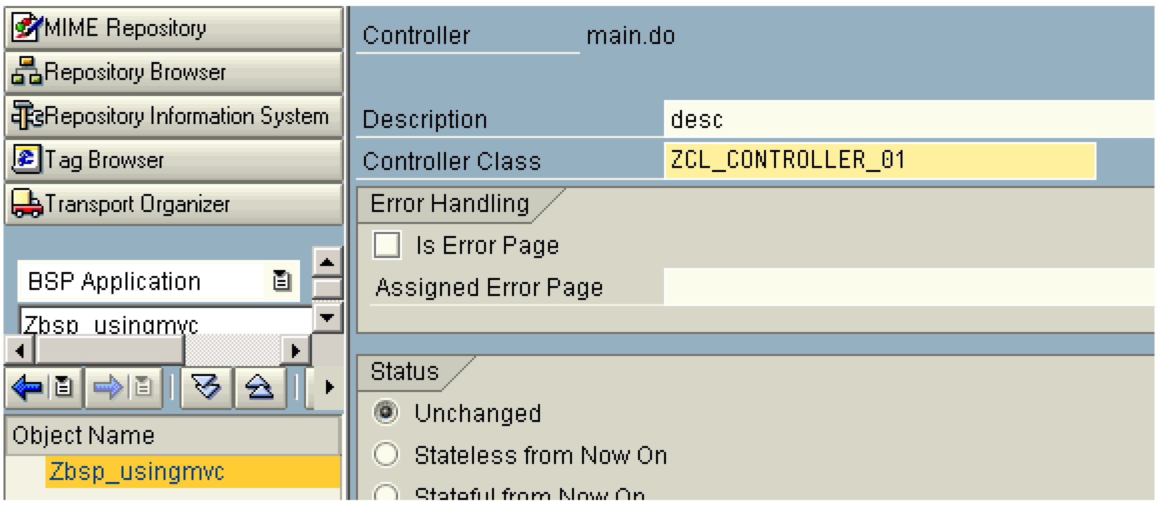


Created By	MISMJL
Last changed by	MISMJL
Package	$TMP
Original Language	EN
Internal name	ZBSP_USINGMV

Step 3 – Populate controller class (zcl_controller_01)

Enter the name of your controller class i.e. zcl_controller_01. We also need to create this class

so double click on the name. Yes.



Controller main.do

Description desc

Controller Class ZCL_CONTROLLER_01

Is Error Page

Assigned Error Page

(●) Unchanged

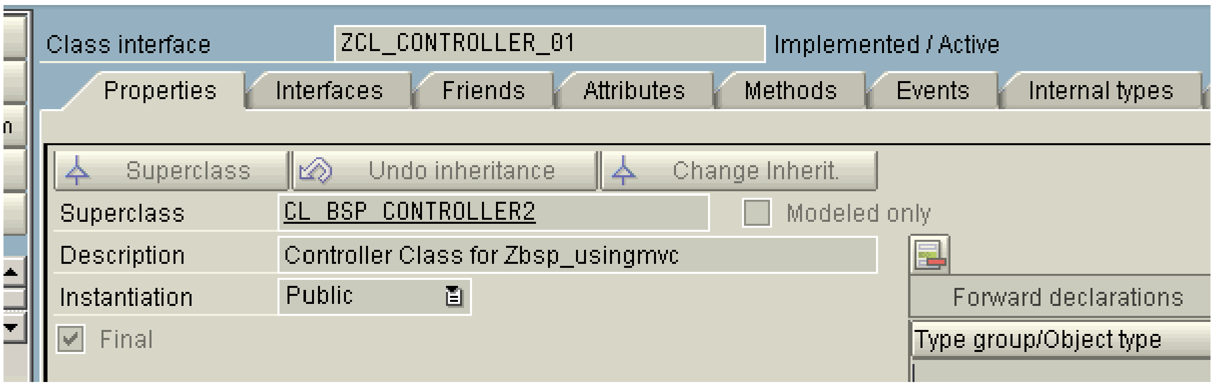
( ) Stateless from Now On

( ) Stateful from Now On

Step 4 – Creating controller class

Once you have double clicked on the controller class name and pressed yes you will need to check the

properties tab and ensure its super class is CL_BSP_CONTROLLER2



Class interface ZCL_CONTROLLER_01 Implemented / Active

Superclass CL_BSP_CONTROLLER2

Modeled only

Description Controller Class for Zbsp_usingmvc

Instantiation Public

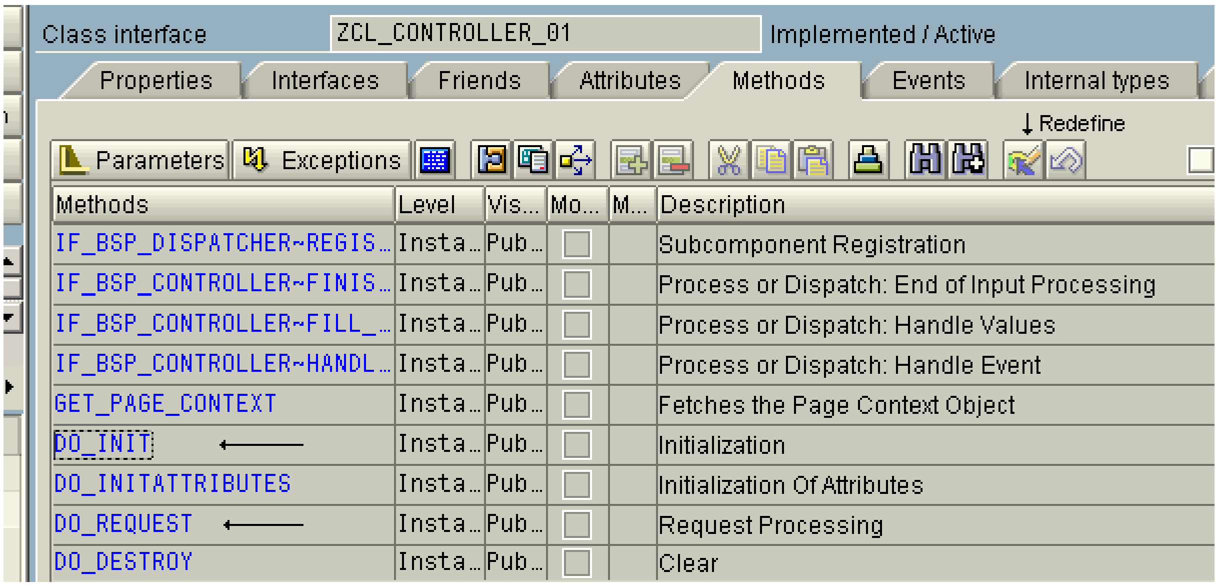
✓ Final

Step 5 – Redefine Methods

You will have inherited a number of methods from the superclass. As these methods work in much the

same way as events do within classic BSPs and ABAP you will now need to redefine a number of these

methods. These are DO_INIT and DO_REQUEST



Class interface ZCL_CONTROLLER_01 Implemented / Active

Methods	Level	Vis...	Mo...	M...	Description
IF_BSP_DISPATCHER~REGIS...	Insta...	Pub...	☐		Subcomponent Registration
IF_BSP_CONTROLLER~FINIS...	Insta...	Pub...	☐		Process or Dispatch: End of Input Processing
IF_BSP_CONTROLLER~FILL_...	Insta...	Pub...	☐		Process or Dispatch: Handle Values
IF_BSP_CONTROLLER~HANDL...	Insta...	Pub...	☐		Process or Dispatch: Handle Event
GET_PAGE_CONTEXT	Insta...	Pub...	☐		Fetches the Page Context Object
DO_INIT ⟵	Insta...	Pub...	☐		Initialization
DO_INITATTRIBUTES	Insta...	Pub...	☐		Initialization Of Attributes
DO_REQUEST ⟵	Insta...	Pub...	☐		Request Processing
DO_DESTROY	Insta...	Pub...	☐		Clear

Step 6 – Redefine DO_INIT

Ensure you have this method available for change by pressing the pencil button. Place your cursor

on the method called DO_INIT and press the redefine button.

For this current example you dont have to do anything in this method but for a follow on exercise

you will be creating the MODEL instance in the DO_INIT method.

Step 7 – Redefine DO_REQUEST(call a layout of type VIEW)

Ensure you have this method available for change (done in previous step). Place your cursor

on the method called DO_REQUEST and press the redefine button. In the DO_request we will call a layout(View). We create a reference variable referencing the page and then call the method

create_view and pass it the actual view (not yet created). We then call the view. Enter the

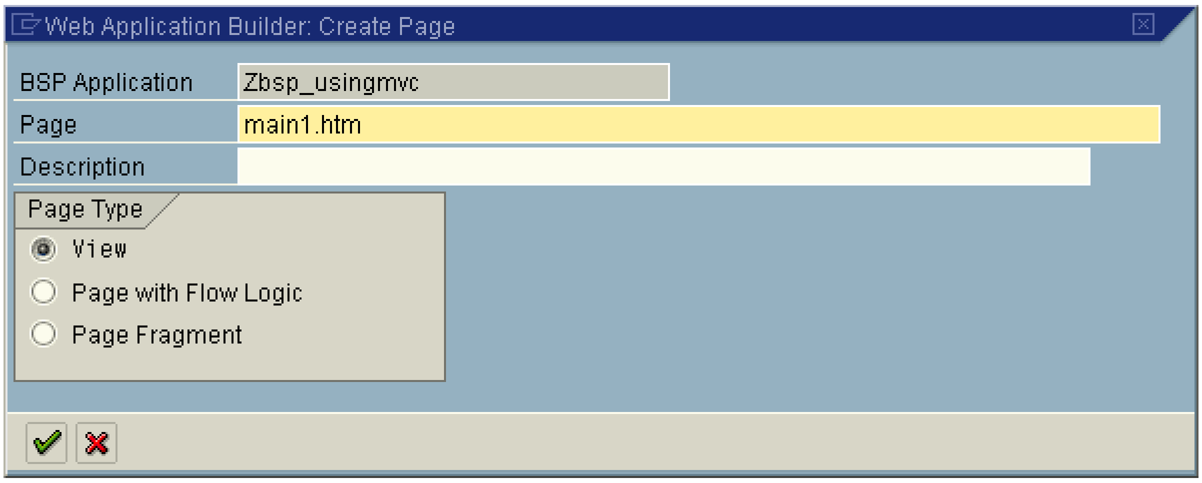
following code:

```
* Create reference variable from the interface if_bsp_page

  DATA: r_view TYPE REF TO if_bsp_page.



* Create object r_view with view_name main1.htm

* Layout is provided by view main1.htm

  r_view = create_view( view_name = 'main1.htm' ).



* Process view-> response is set accordingly

  call_view( r_view ).
```

Step 8 – Create BSP Page (View)

Right click the BSP Application and create a new page called main1.htm. When this is done

make sure you save and activate it.


Web Application Builder: Create Page

BSP Application	Zbsp_usingmvc
Page	main1.htm
Description	

Page Type
[✓ View]
Page with Flow Logic
Page Fragment

Step 9 – Activate whole BSP Application

Tutorial 2 - Creating the Model (Class to to perform functionality i.e. retrieve data)

Step 1 - Using the Model class within DO_INIT (note: Model class not created yet!)

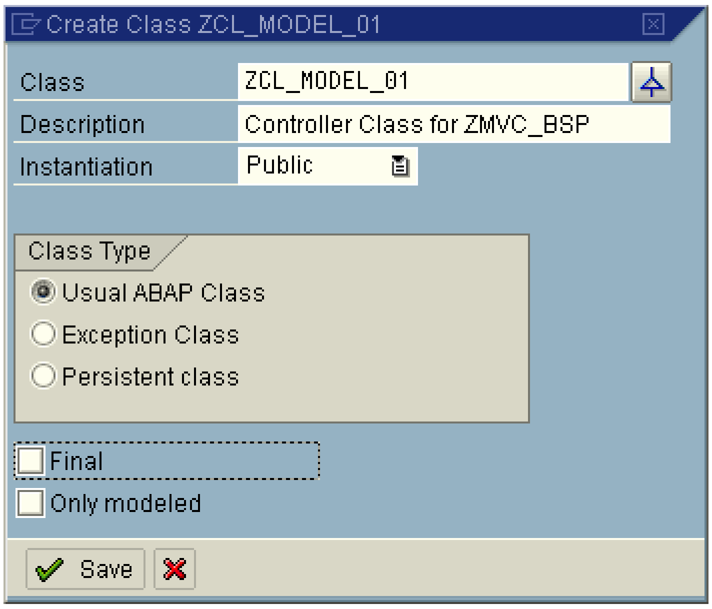
From within SE80 double click on main.do to select it, now double click on the controller class.

Double click on the DO_INIT method. Now enter the following code into the DO_INIT method & save.

```
method DO_INIT.
*CALL METHOD SUPER->DO_INIT
*    .
* Create refernece variable based on your own class (not created yet)
data: r_model TYPE REF TO zcl_model_01.

* Create an instance of our Model class and use a widening cast to
load your
* reference variable r_model
r_model ?= me->create_model(
       class_name = 'ZCL_MODEL_01'
       model_id   = 'mod_main' ).

* Use the r_model to call the select details method from your Model
class
r_model->select_details( ).
* Load attributes in your class attributes to hold the variable - make
it
* more 'global' so it can be seen by other methods.
me->r_model = r_model.
endmethod.
```

Step 2 – Create model class

Use SE80 or SE18 to create a new class, give it a name and description.


Create Class ZCL_MODEL_01

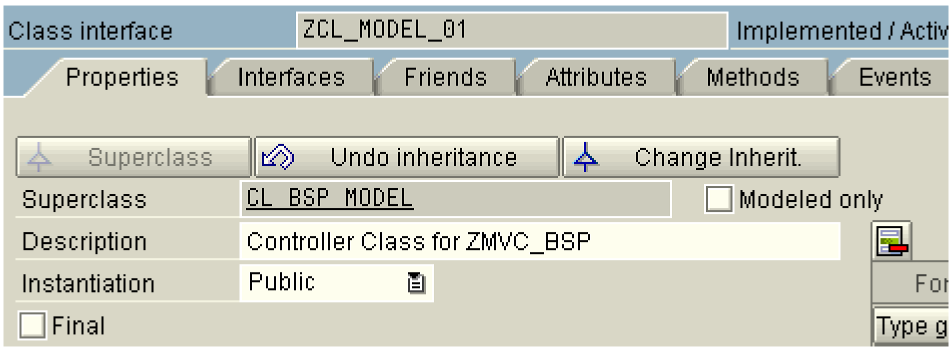
Class	ZCL_MODEL_01
Description	Controller Class for ZMVC_BSP
Instantiation	Public

Class Type
[✓ Usual ABAP Class]
Exception Class
Persistent class

Final
Only modeled

Go to the properties tab and enter change mode, Press the Superclass button and enter the

superclass cl_bsp_model. Save and activate


Class interface ZCL_MODEL_01 Implemented / Activ

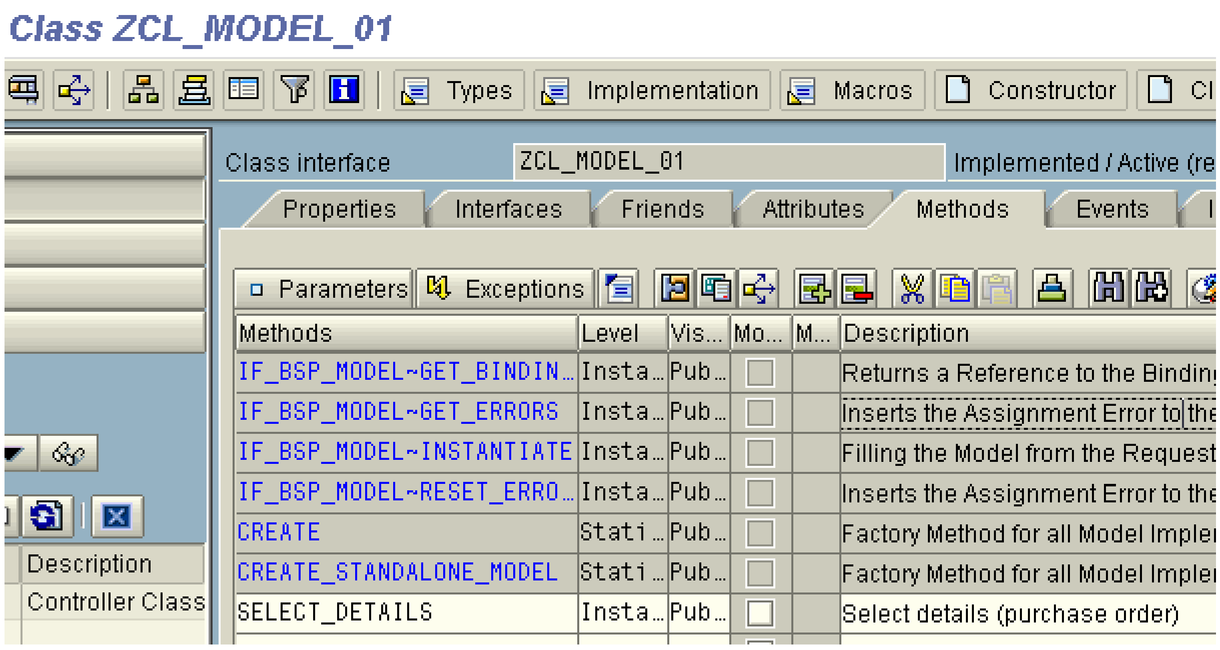
Superclass	CL_BSP_MODEL
Description	Controller Class for ZMVC_BSP
Instantiation	Public

Modeled only
Final

Step 3 – Define method of Model class

Select the methods tab and scroll to the bottom of the methods, now enter a new method called

SELECT_DETAILS, as an instance method with public visibility.


Class ZCL_MODEL_01

Class interface ZCL_MODEL_01 Implemented / Active (re

Methods	Level	Vis...	Mo...	M...	Description
IF_BSP_MODEL~GET_BINDIN...	Insta...	Pub...	☐		Returns a Reference to the Bindin
IF_BSP_MODEL~GET_ERRORS	Insta...	Pub...	☐		Inserts the Assignment Error to the
IF_BSP_MODEL~INSTANTIATE	Insta...	Pub...	☐		Filling the Model from the Request
IF_BSP_MODEL~RESET_ERRO...	Insta...	Pub...	☐		Inserts the Assignment Error to the
CREATE	Stati...	Pub...	☐		Factory Method for all Model Imple
CREATE_STANDALONE_MODEL	Stati...	Pub...	☐		Factory Method for all Model Imple
SELECT_DETAILS	Insta...	Pub...	☐		Select details (purchase order)

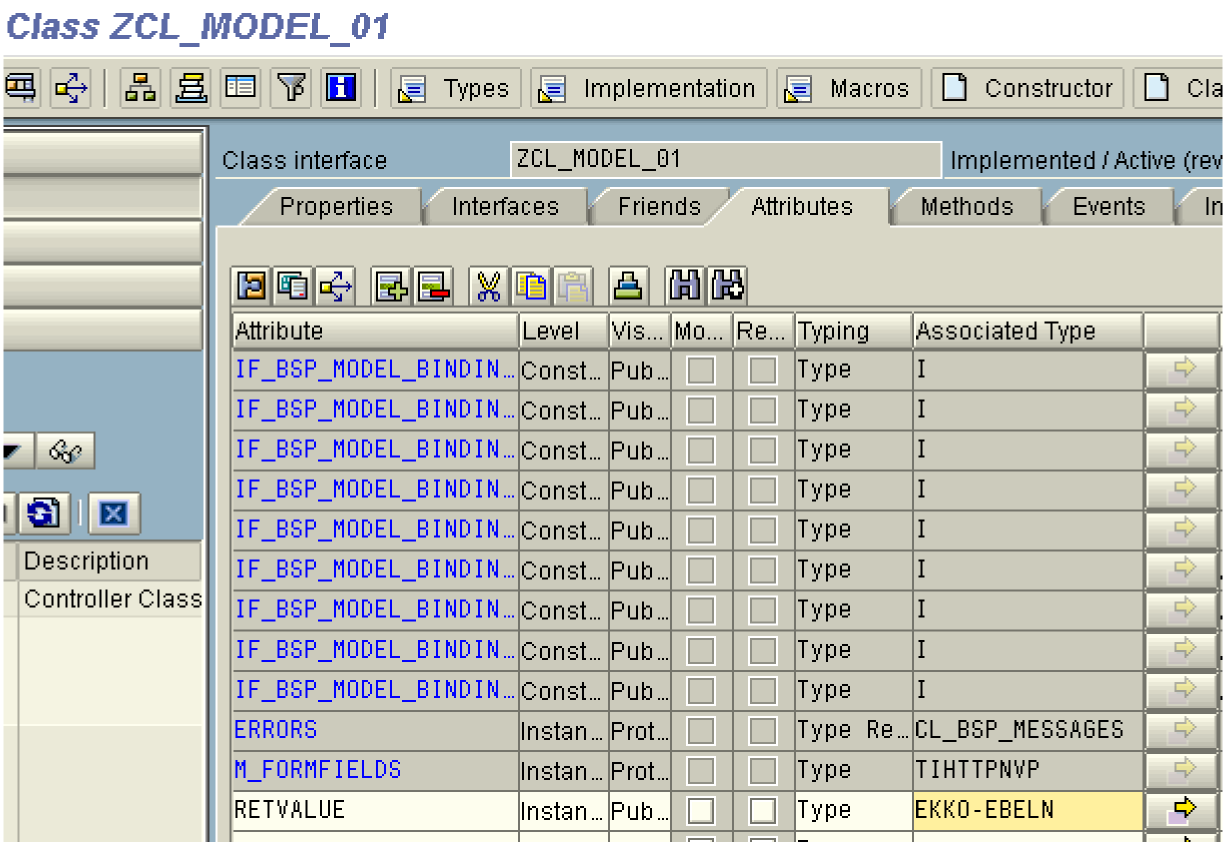
Now double click the method to create it and enter the following code:

```
METHOD select_details .
  SELECT ebeln
   UP TO 1 ROWS
    INTO retvalue
    FROM ekko.
  ENDSELECT.
ENDMETHOD.
```

Step 4 – Define attributes of method

Click on the Model class attributes tab and enter the field 'RETVALUE' as type ekko-ebeln, Ensuring it

is an instance attribute, which has public visibility. Now save and activate the new model class!



Class interface ZCL_MODEL_01 Implemented / Active (rev

Attribute	Level	Vis...	Mo...	Re...	Typing	Associated Type	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
IF_BSP_MODEL_BINDIN...	Const...	Pub...	☐	☐	Type	I	
ERRORS	Instan...	Prot...	☐	☐	Type Re...	CL_BSP_MESSAGES	
M_FORMFIELDS	Instan...	Prot...	☐	☐	Type	TIHTTPNVP	
RETVALUE	Instan...	Pub...	☐	☐	Type	EKKO-EBELN	

Step 5 – Define atributes of the controller sub class

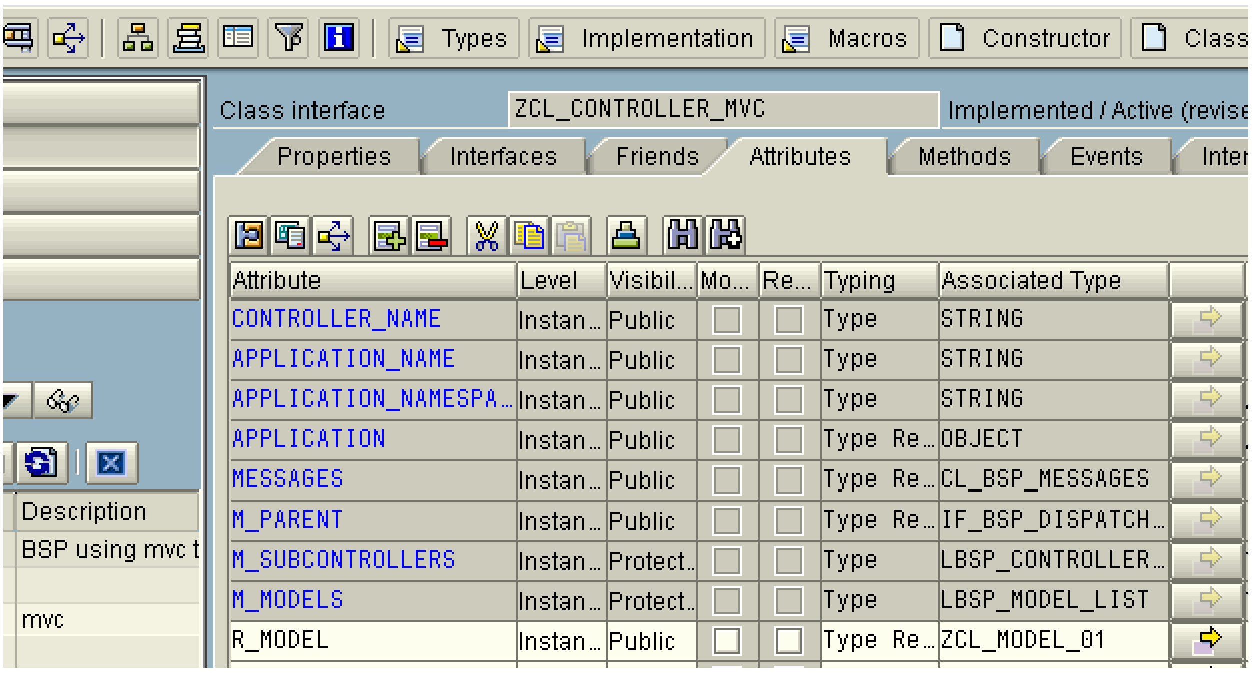
Now return to the controller class you created, accessed via the controller page (i.e. main.do). Remember

the ABAP code you inserted to declare 'r_model' within the DO_INIT method? You now need to declare this

attribute within the class attributes tab. It needs to be instance, public and 'type ref to' your model

class (ZCL_MODEL_01). Save and Activate the controller class (ZCL_CONTROLLER_01).

Class ZCL_CONTROLLER_MVC



Class interface ZCL_CONTROLLER_MVC Implemented / Active (revise

BSP using mvc t

mvc

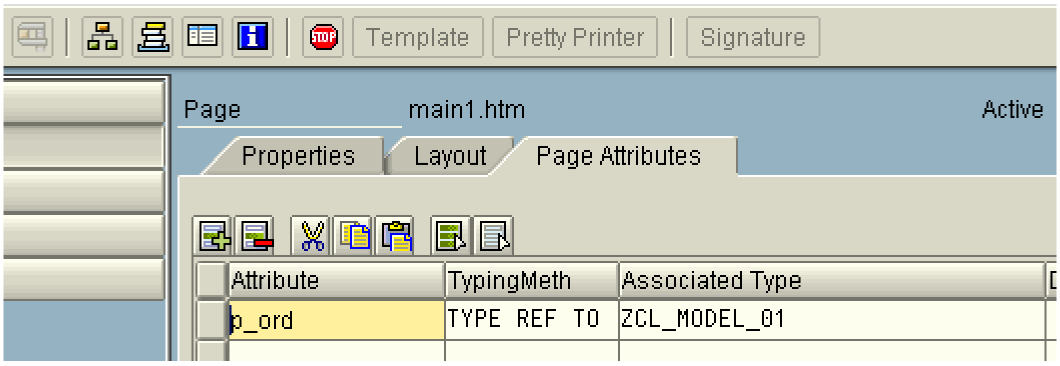
Attribute	Level	Visibil...	Mo...	Re...	Typing	Associated Type
CONTROLLER_NAME	Instan...	Public	☐	☐	Type	STRING
APPLICATION_NAME	Instan...	Public	☐	☐	Type	STRING
APPLICATION_NAMESPA...	Instan...	Public	☐	☐	Type	STRING
APPLICATION	Instan...	Public	☐	☐	Type Re...	OBJECT
MESSAGES	Instan...	Public	☐	☐	Type Re...	CL_BSP_MESSAGES
M_PARENT	Instan...	Public	☐	☐	Type Re...	IF_BSP_DISPATCH...
M_SUBCONTROLLERS	Instan...	Protect...	☐	☐	Type	LBSP_CONTROLLER...
M_MODELS	Instan...	Protect...	☐	☐	Type	LBSP_MODEL_LIST
R_MODEL	Instan...	Public	☐	☐	Type Re...	ZCL_MODEL_01

Step 6 – Display data from the model (update the page/view)

In-order to display the data from the model, we are going to use a reference variable p_ord declared in

the page attributes .

Edit Page ZMVC_BSP



Page main1.htm Active

Attribute	TypingMeth	Associated Type
p_ord	TYPE REF TO	ZCL_MODEL_01

Now make changes to the layout, so that the returned data is displayed within and input field.

```
<%@page language="abap"%>
```

```
<%@extension name="htmlb" prefix="htmlb"%>

<htmlb:content design="design2003">
  <htmlb:page title = " ">
    <htmlb:form>
      <htmlb:textView         text          = "Purchase order"
                              design        = "EMPHASIZED" />
      <htmlb:inputField       id            = ""
                              invalid       = "false"
                              value         = "test"
                              required      = "true"/><BR>
      <htmlb:button           text          = "Press Me"
                              onClick       = "myClickHandler" />

    </htmlb:form>
  </htmlb:page>
</htmlb:content>
```

Step 7 – Display data from the model (update controller)

Within the DO_REQUEST of the controller class (ZCL_CONTROLLER_01) enter the code below to pass the

model reference back to the View. Save and activate everything.

```
METHOD do_request .

*CALL METHOD SUPER->DO_REQUEST

*    .

  DATA: r_view TYPE REF TO if_bsp_page.

  r_view = create_view( view_name = 'main1.htm' ).

  r_view->set_attribute( name  = 'p_ord'

                       value = me->r_model ).

  call_view( r_view ).

ENDMETHOD.
```

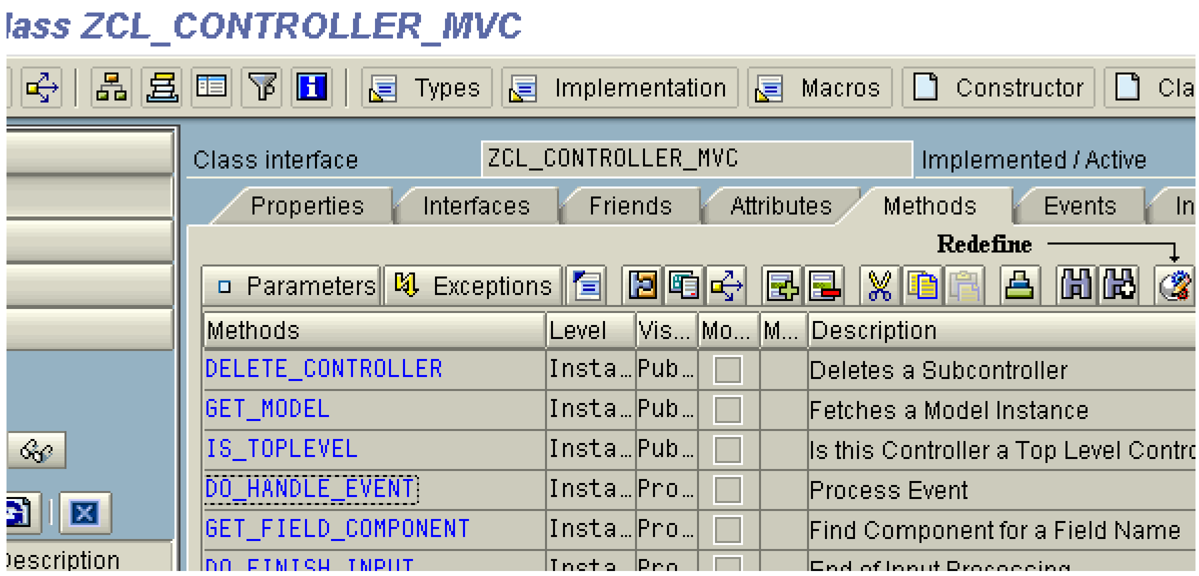
Tutorial 3 - Event handling and calling a new view

Step 1 - Redefine DO_HANDLE_EVENT event

Return to the controller class you created in tutorial 1, accessed via the controller page (i.e. main.do)

and double clicking on the cc name (ZCL_CONTROLLER_01). Go into change mode and find the

DO_HANDLE_EVENT method and redefine it.



Class interface ZCL_CONTROLLER_MVC Implemented / Active

Methods	Level	Vis...	Mo...	M...	Description
DELETE_CONTROLLER	Insta...	Pub...	☐		Deletes a Subcontroller
GET_MODEL	Insta...	Pub...	☐		Fetches a Model Instance
IS_TOPLEVEL	Insta...	Pub...	☐		Is this Controller a Top Level Contro
DO_HANDLE_EVENT	Insta...	Pro...	☐		Process Event
GET_FIELD_COMPONENT	Insta...	Pro...	☐		Find Component for a Field Name
DO_FINISH_INPUT	Insta...	Pro...	☐		End of Input Processing

Step 2 - Insert code in to DO_HANDLE_EVENT

Enter the following ABAP code which handles a button click event:

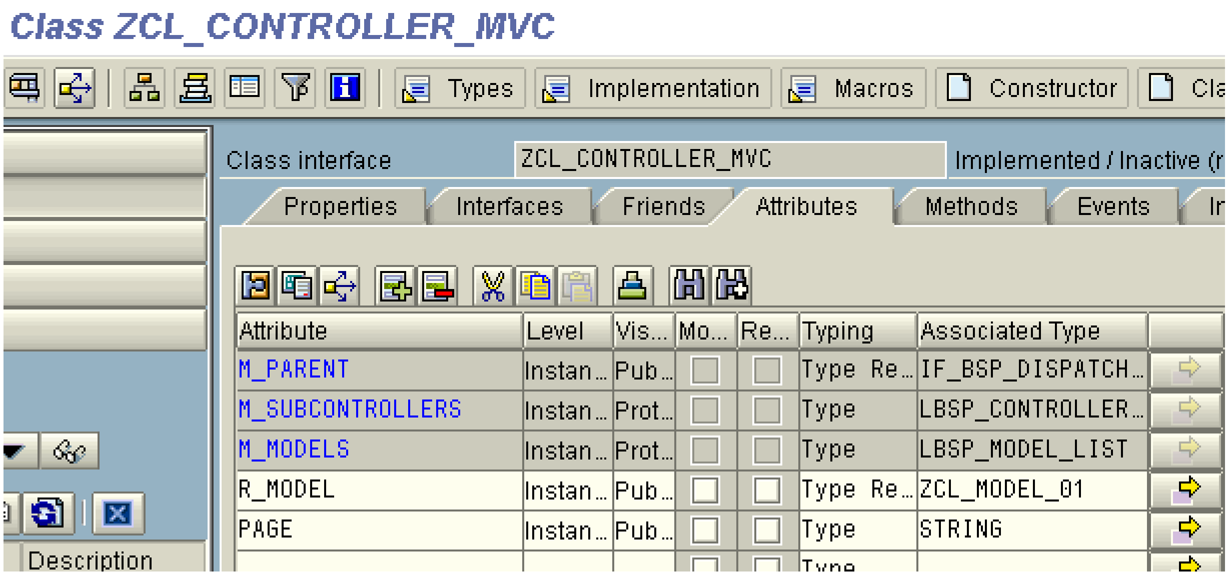
```
method DO_HANDLE_EVENT .
*CALL METHOD SUPER->DO_HANDLE_EVENT
*   EXPORTING
*     EVENT            =
*     HTMLB_EVENT      =
**     HTMLB_EVENT_EX   =
*     GLOBAL_MESSAGES =
*   RECEIVING
*     GLOBAL_EVENT     =
*     .
  DATA: button_event TYPE REF TO CL_HTMLB_EVENT_BUTTON.      "date
event
  DATA: date_event TYPE REF TO CL_HTMLB_EVENT_DATENAVIGATOR. "button
event

* Check if event being processed is a button event
```

```
  IF htmlb_event IS BOUND AND htmlb_event->name = 'button'.
*   Use widening cast to take generic event to specific event (button
event)
*    - Basically moves current event structure into button event
structure,
*    - so that the button event now contains the relevant data
    button_event ?= htmlb_event.
*
*   Contains value store in the 'onClick' parameter on page view
    if button_event->server_event = 'myClickHandler'.
        page = 'page2.htm'.
    endif.
  ENDIF.

* Check if event being processed is a date event
* - the below code is simply for further demonstration of above syntax
  IF htmlb_event IS BOUND AND htmlb_event->name = 'dateNavigator'.
    date_event ?= htmlb_event.
  ENDIF.
endmethod.
```

Step 3 – Create attributte to store next page value

Return back to Class interface and define a new class attributte as type string to store next page

value!


Class ZCL_CONTROLLER_MVC

Class interface ZCL_CONTROLLER_MVC Implemented / Inactive

Attribute	Level	Vis...	Mo...	Re...	Typing	Associated Type
M_PARENT	Instan...	Pub...	☐	☐	Type Re...	IF_BSP_DISPATCH...
M_SUBCONTROLLERS	Instan...	Prot...	☐	☐	Type	LBSP_CONTROLLER...
M_MODELS	Instan...	Prot...	☐	☐	Type	LBSP_MODEL_LIST
R_MODEL	Instan...	Pub...	☐	☐	Type Re...	ZCL_MODEL_01
PAGE	Instan...	Pub...	☐	☐	Type	STRING
			☐	☐	Type	

Step 4 – Modify DO_REQUEST method

You now need to modify the DO_REQUEST code so that it calls the event handling and controls which page to display based on the new page variable/attribute. The event handling is called using the

'dispatch_input( )' command.

```
METHOD do_request .
*CALL METHOD SUPER->DO_REQUEST
*    .
  DATA: r_view TYPE REF TO if_bsp_page.

* Calls event handler DO_HANDLE_EVENT
  dispatch_input( ).

  IF page EQ 'main1.htm' or page EQ space.
    r_view = create_view( view_name = 'main1.htm' ).

    r_view->set_attribute( name = 'p_ord'
                           value = me->r_model ).
  ELSEIF page = 'page2.htm'.
    r_view = create_view( view_name = 'page2.htm' ).

    r_view->set_attribute( name = 'p_ord'
                           value = me->r_model ).
  ENDIF.

** Create object r_view with view_name main1.htm
** Layout is provided by view main1.htm
*  r_view = create_view( view_name = 'main1.htm' ).
*  r_view->set_attribute( name  = 'p_ord'
*                         value = me->r_model ).
  call_view( r_view ).
ENDMETHOD.
```

Step 5 – Create second View

Firstly save and activate the controller class. The next stage is to create the second view which

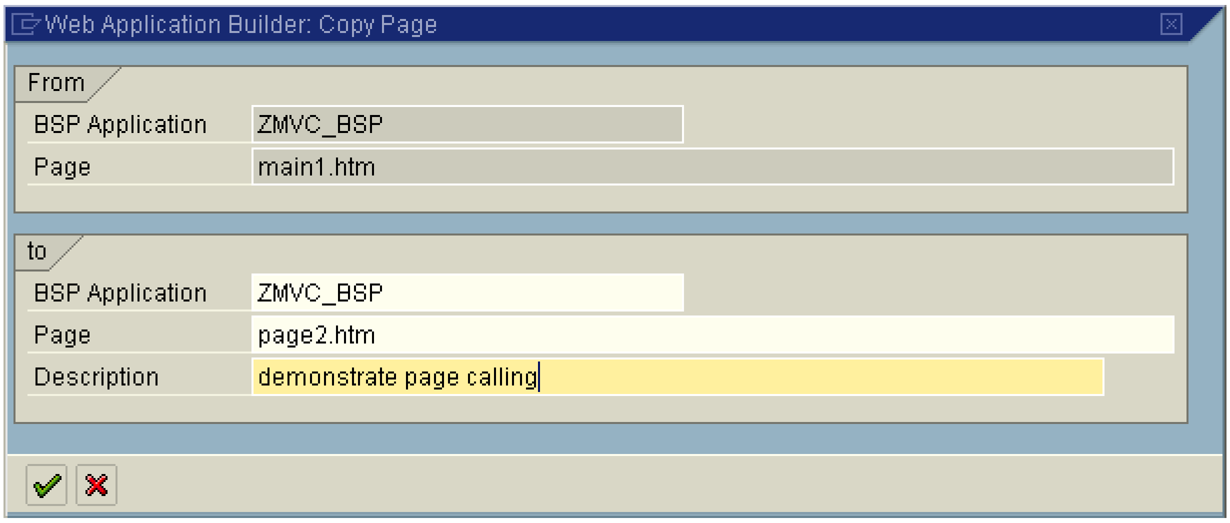
is executed from within DO_REQUEST. This will need to be called 'page2.htm' unless you modify the

code you have just placed in the DO_REQUEST method. The simplest way to do this is to copy your

existing view (main1.htm), You might want to change some text slightly so that you can distanguish

between the 2 page.

i.e. change PO text to 'Purchase order2'.


Web Application Builder: Copy Page

From

BSP Application	ZMVC_BSP
Page	main1.htm

to

BSP Application	ZMVC_BSP
Page	page2.htm
Description	demonstrate page calling

Step 6 – Save and activate

Ensure you save and activate all the objects that have been changed during this tutorial.