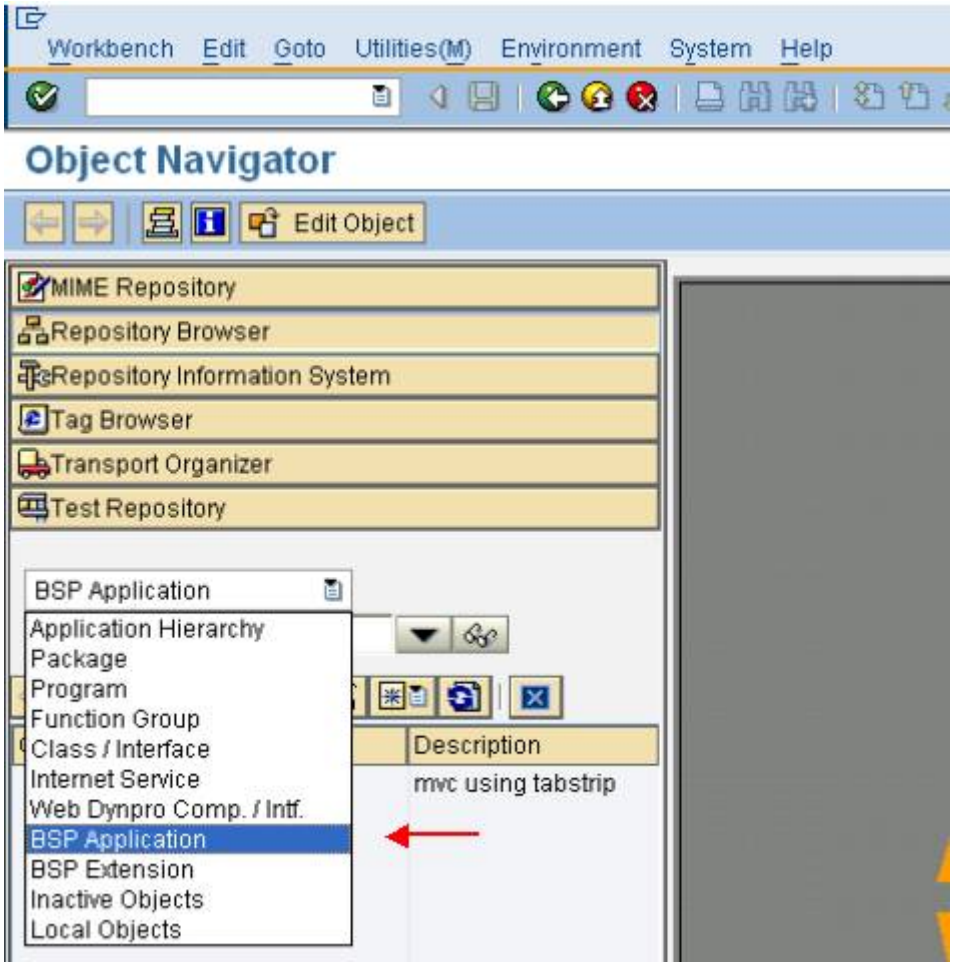
BSP Application using MVC Architecture - Displaying Business Partner data using a BAPI

By Raghava Vakada, Mouri Tech Solutions

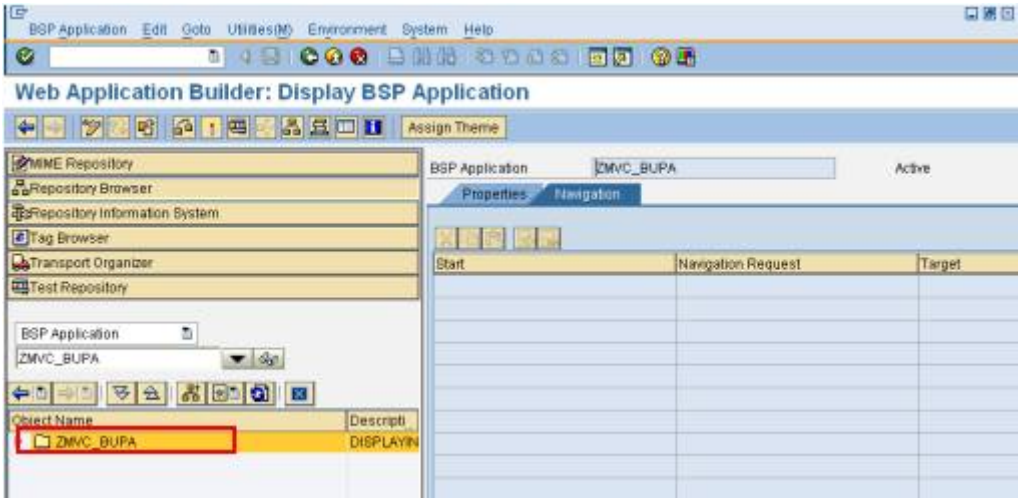
Requirement: To display business partner data using the BAPI bapi_bupa_search_2 on a BSP application view.

STEPS:

1) Go to transaction SE80, select 'BSP application' from the dropdown list.



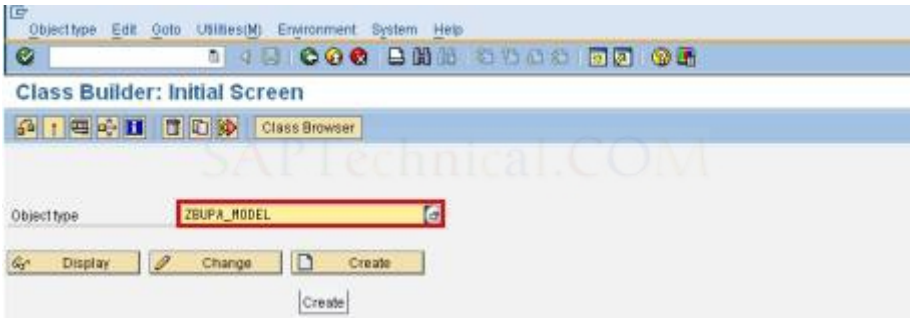
2) Create a BSP application with the name ZMVC_BUPA.



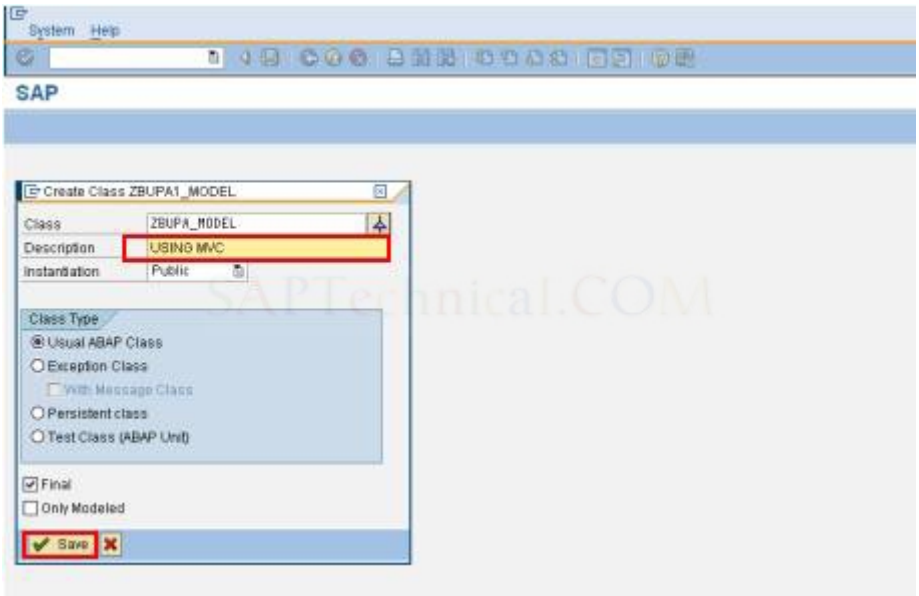
3) Creating a Model Class:

Go to transaction SE24 and enter the name of the class as shown below.

Class name: ZBUPA_MODEL

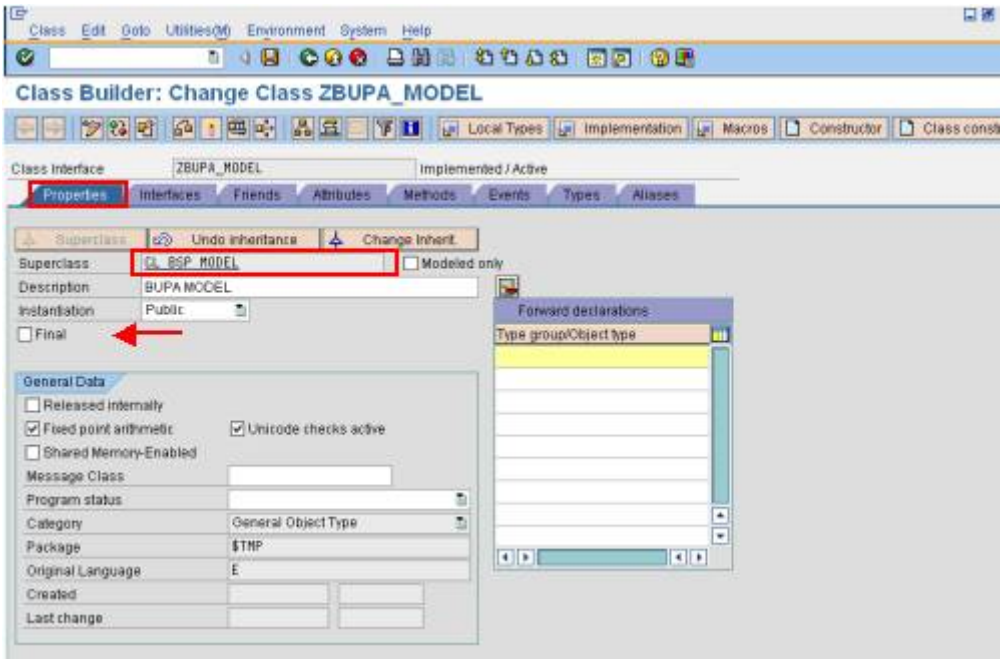


In the pop up window that appears, enter the description and then click on the ‘save’ button.

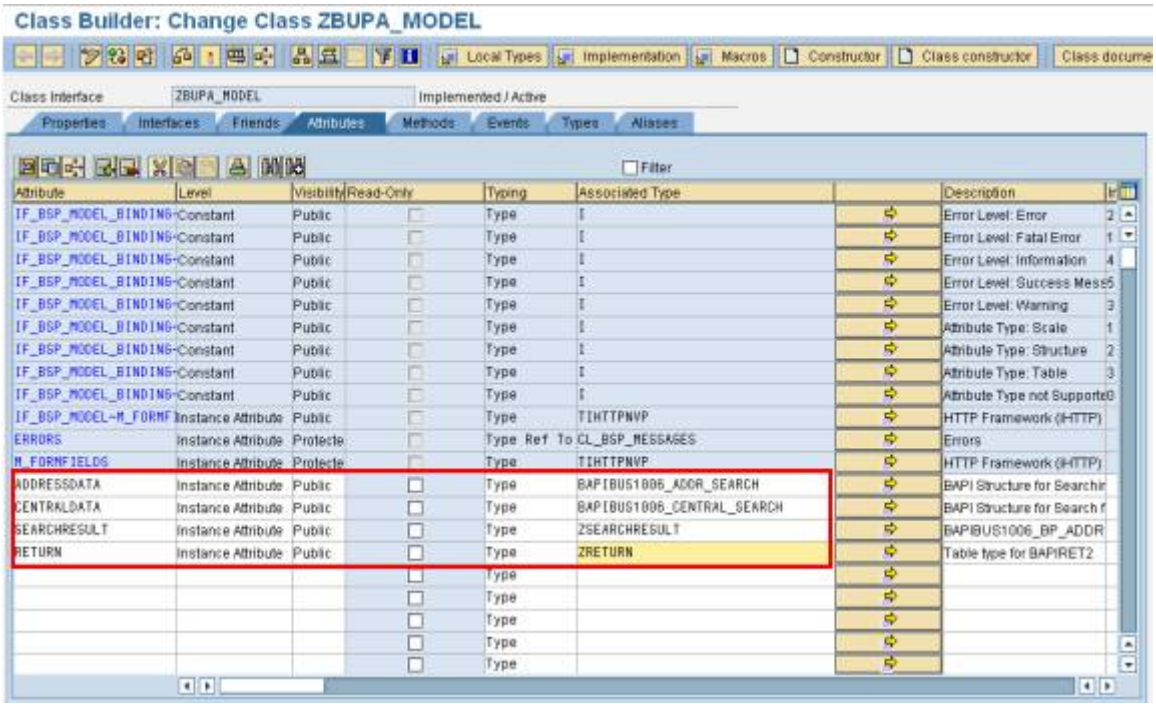


In the class, go to the properties tab and then specify the superclass for the new model class so that all the methods and attributes will be inherited to the model class.

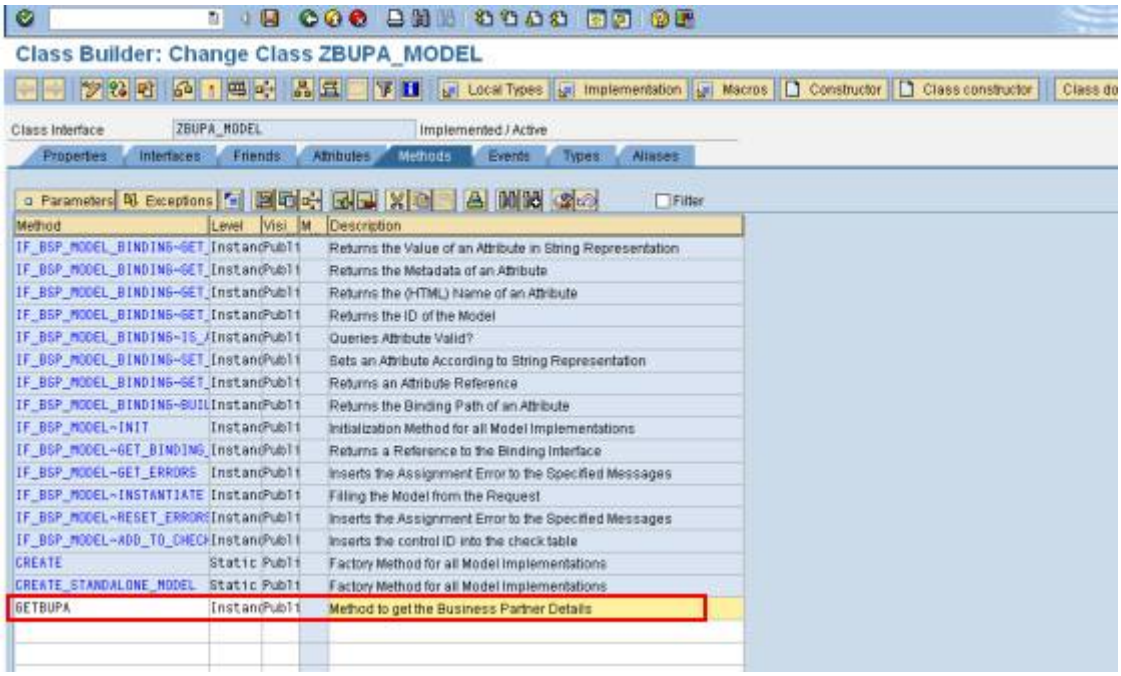
You can uncheck the checkbox ‘Final’.



Add attributes to the new model class. Please create table types for those attributes in the tables tab of the BAPI. Here, you can see that I have used the table types to refer to the attributes of our class.



Add a method to the new model class to get the business partner details

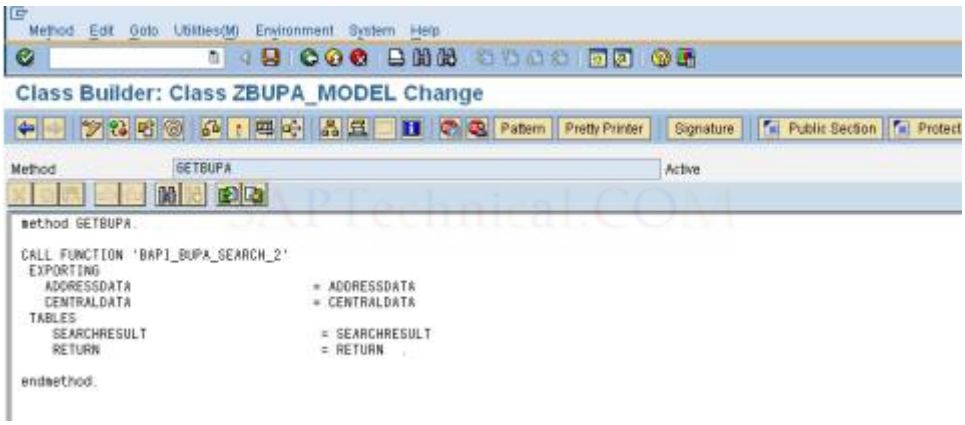


Modified method implementations

Double click on the GETBUPA method.

Write the following code.

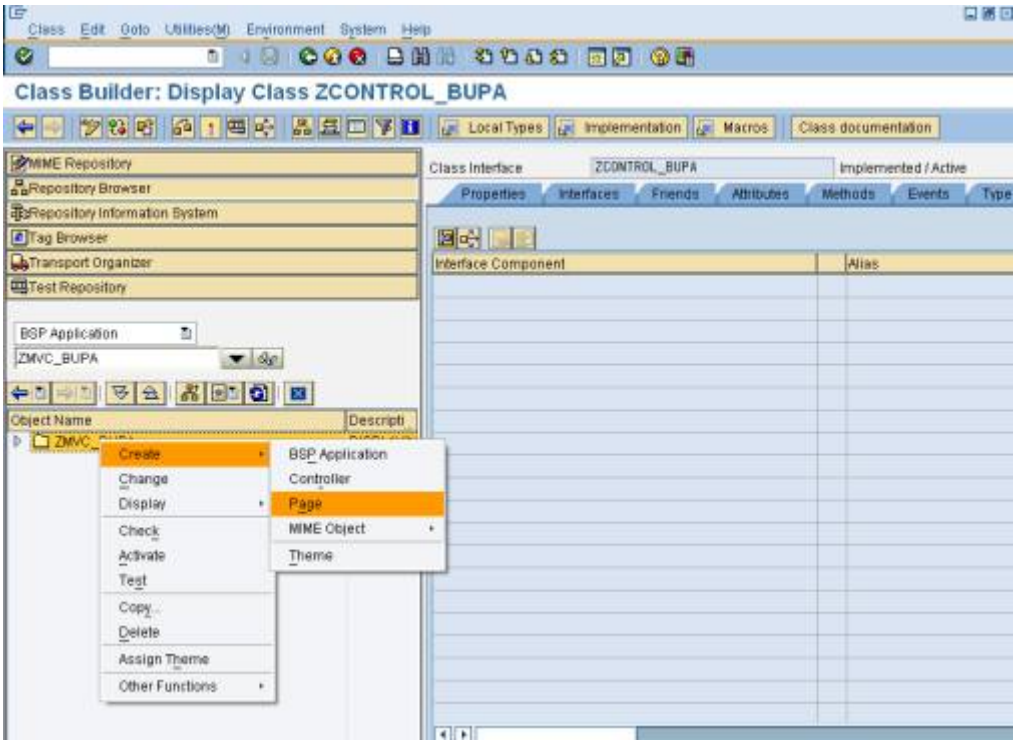
```
Method GETBUPA.
CALL FUNCTION 'BAPI_BUPA_SEARCH_2'
 EXPORTING
   ADDRESSDATA                    = ADDRESSDATA
   CENTRALDATA                    = CENTRALDATA
 TABLES
    SEARCHRESULT                   = SEARCHRESULT
    RETURN                         = RETURN.
endmethod.
```



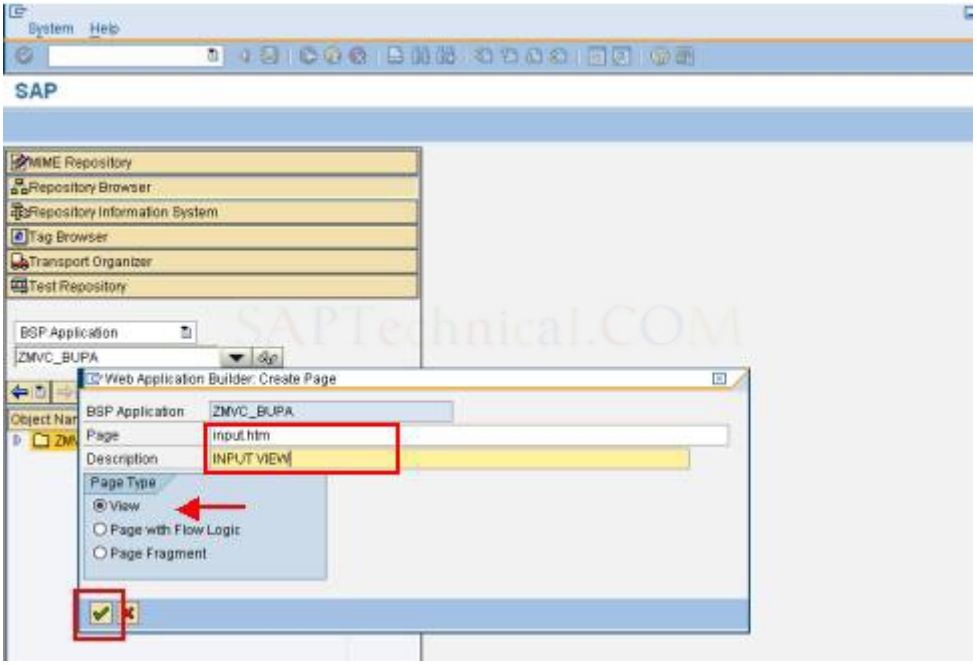
Save, check and activate the class.

CREATING VIEWS.

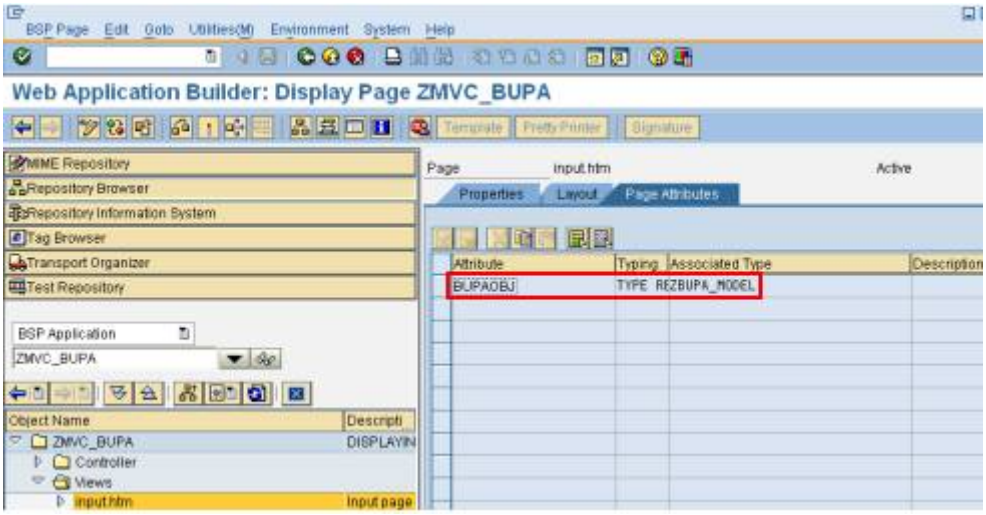
1) Create view: In transaction SE80, Right-click on the BSP application name and navigate to Create→Page.



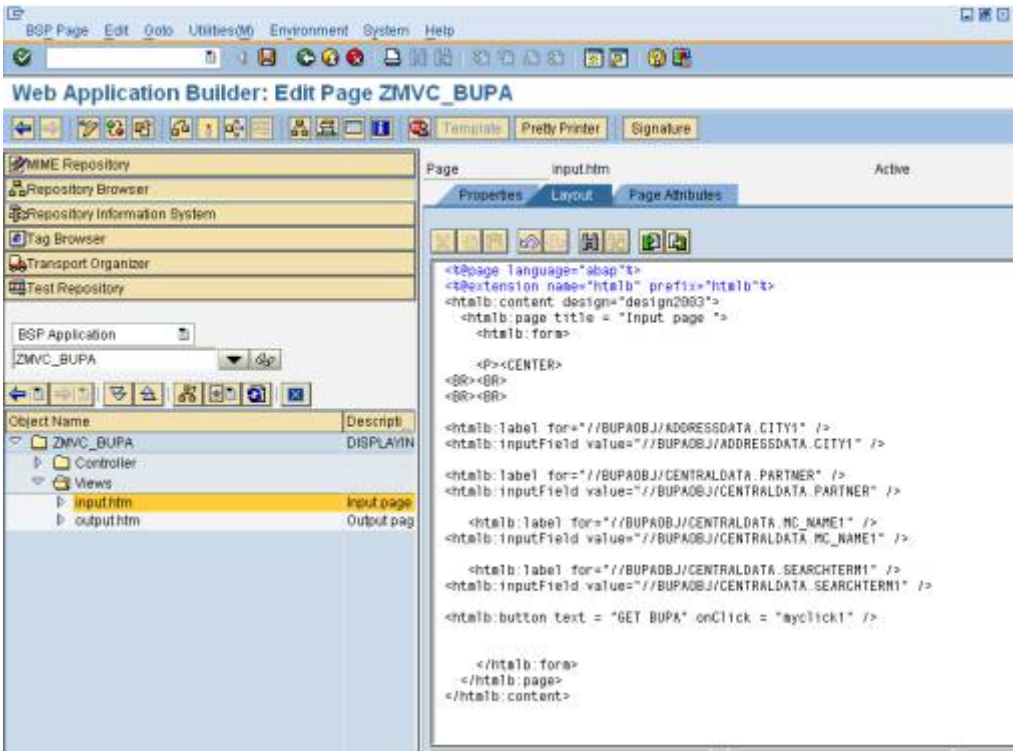
As shown below, enter the name and description of the view that you want to create. Select the radio button 'View', and then click on the 'Continue' button.



Give the page attributes of the view. The attribute of the view is an object reference of the model class that we created previously.



Modify layout code for the search view (input.htm) as shown below.

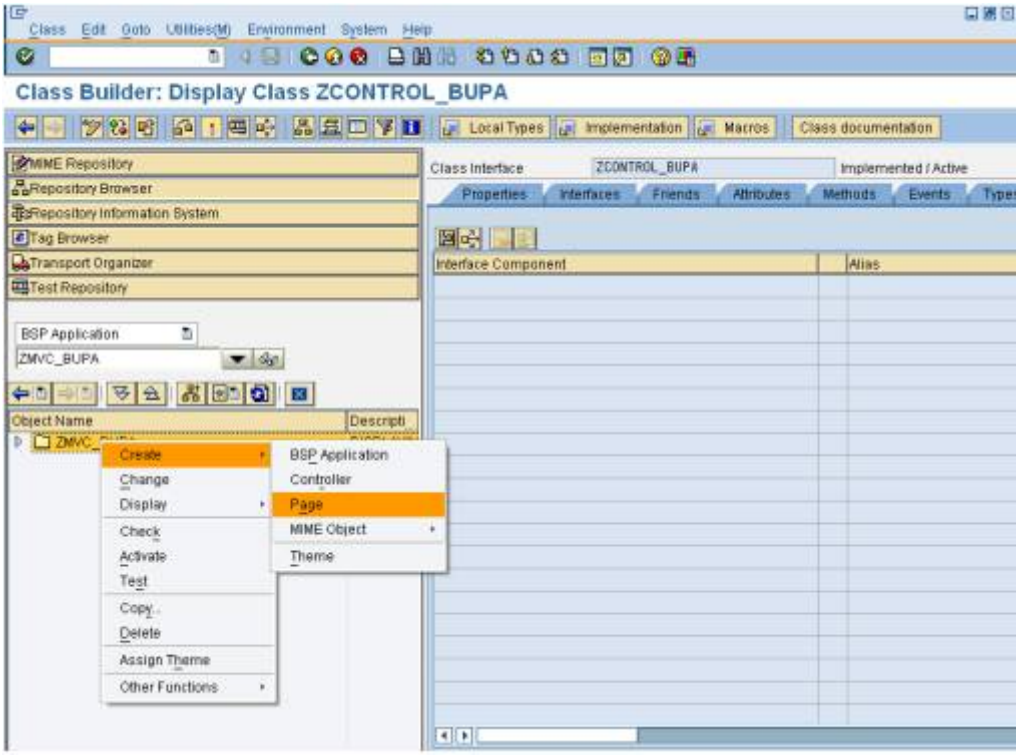


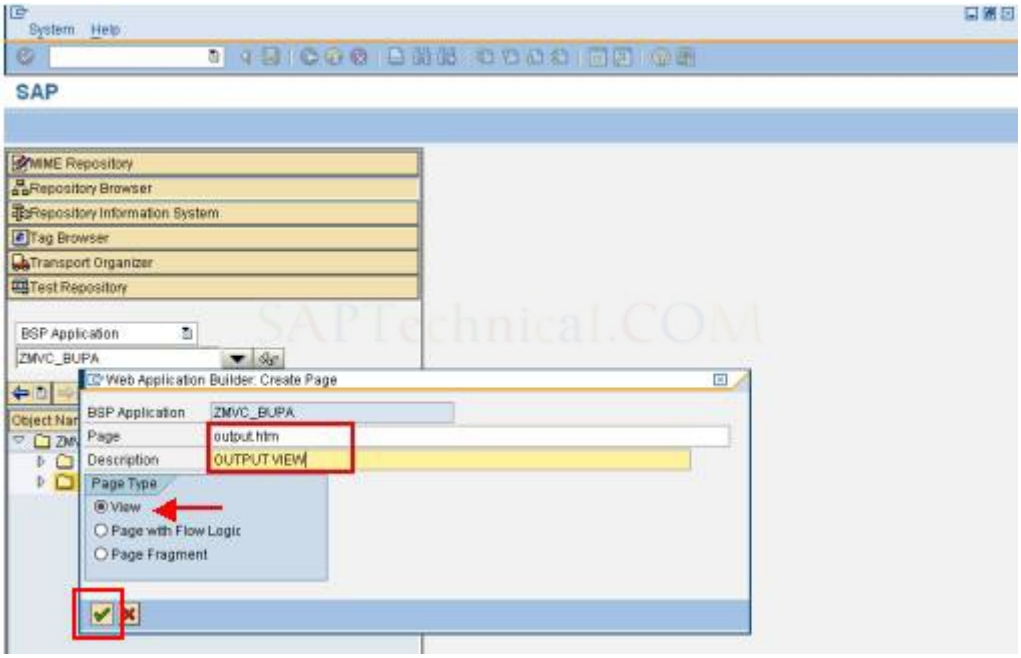
Code :

```
Code :
<%@page language="abap"%>
<%@extension name="htmlb" prefix="htmlb"%>
<htmlb:content design="design2003">
  <htmlb:page title = "Input page ">
    <htmlb:form>
    <P><CENTER>
<BR><BR>
```

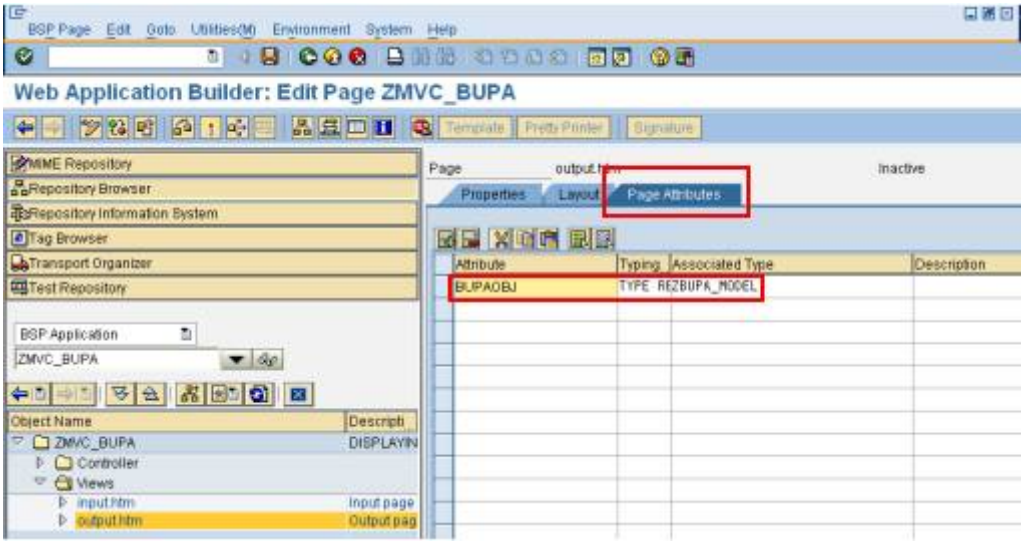
```
<BR><BR>
<htmlb:label for="//BUPAOBJ/ADDRESSDATA.CITY1" />
<htmlb:inputField value="//BUPAOBJ/ADDRESSDATA.CITY1" />
<htmlb:label for="//BUPAOBJ/CENTRALDATA.PARTNER" />
<htmlb:inputField value="//BUPAOBJ/CENTRALDATA.PARTNER" />
   <htmlb:label for="//BUPAOBJ/CENTRALDATA.MC_NAME1" />
<htmlb:inputField value="//BUPAOBJ/CENTRALDATA.MC_NAME1" />
   <htmlb:label for="//BUPAOBJ/CENTRALDATA.SEARCHTERM1" />
<htmlb:inputField value="//BUPAOBJ/CENTRALDATA.SEARCHTERM1" />
<htmlb:button text = "GET BUPA" onClick = "myclick1" />
    </htmlb:form>
  </htmlb:page>
</htmlb:content>
```

2) Create second view.

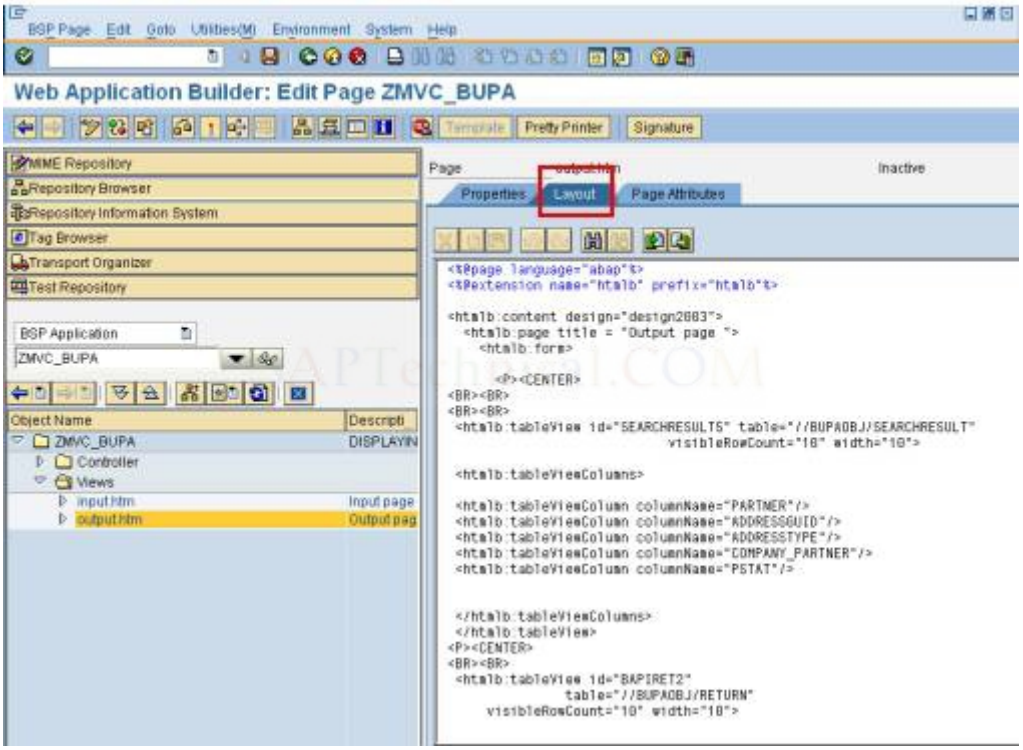




Give the page attributes of the view. The attribute of the view is an object reference of the model class that we created previously.



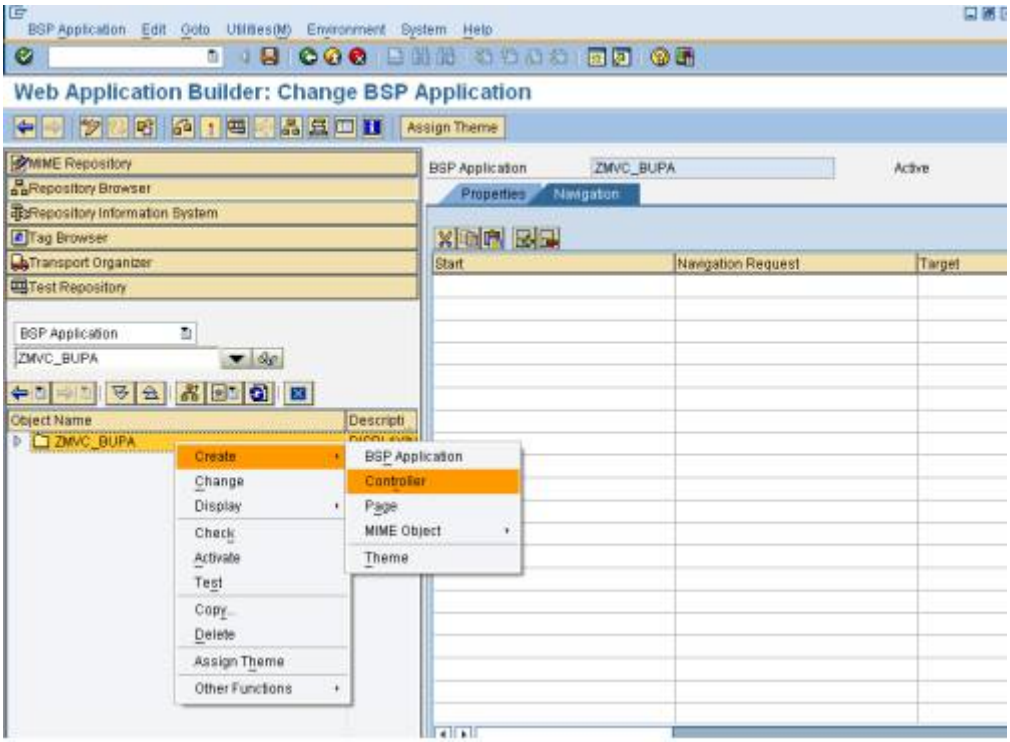
Modify layout code for the search view (output.htm) as shown below.



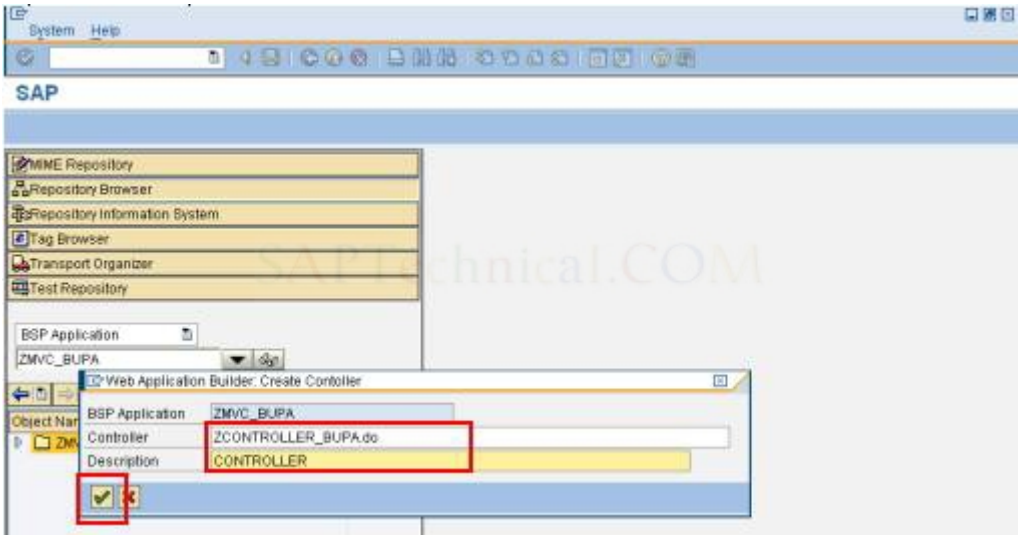
Code :

```
<%@page language="abap"%>
<%@extension name="htmlb" prefix="htmlb"%>
<htmlb:content design="design2003">
  <htmlb:page title = "Output page ">
    <htmlb:form>
      <P><CENTER>
<BR><BR>
<BR><BR>
 <htmlb:tableView id="SEARCHRESULTS" table="//BUPAOBJ/SEARCHRESULT"
                            visibleRowCount="10" width="10">
 <htmlb:tableViewColumns>
 <htmlb:tableViewColumn columnName="PARTNER"/>
 <htmlb:tableViewColumn columnName="ADDRESSGUID"/>
 <htmlb:tableViewColumn columnName="ADDRESSTYPE"/>
 <htmlb:tableViewColumn columnName="COMPANY_PARTNER"/>
 <htmlb:tableViewColumn columnName="PSTAT"/>
 </htmlb:tableViewColumns>
 </htmlb:tableView>
<P><CENTER>
<BR><BR>
 <htmlb:tableView id="BAPIRET2"
                table="//BUPAOBJ/RETURN"
     visibleRowCount="10" width="10">
 <htmlb:tableViewColumns>
 <htmlb:tableViewColumn columnName="TYPE"/>
 <htmlb:tableViewColumn columnName="MESSAGE"/>
 <htmlb:tableViewColumn columnName="PARAMETER"/>
 <htmlb:tableViewColumn columnName="SYSTEM"/>
 </htmlb:tableViewColumns>
 </htmlb:tableView>
    </htmlb:form>
  </htmlb:page>
</htmlb:content>
```

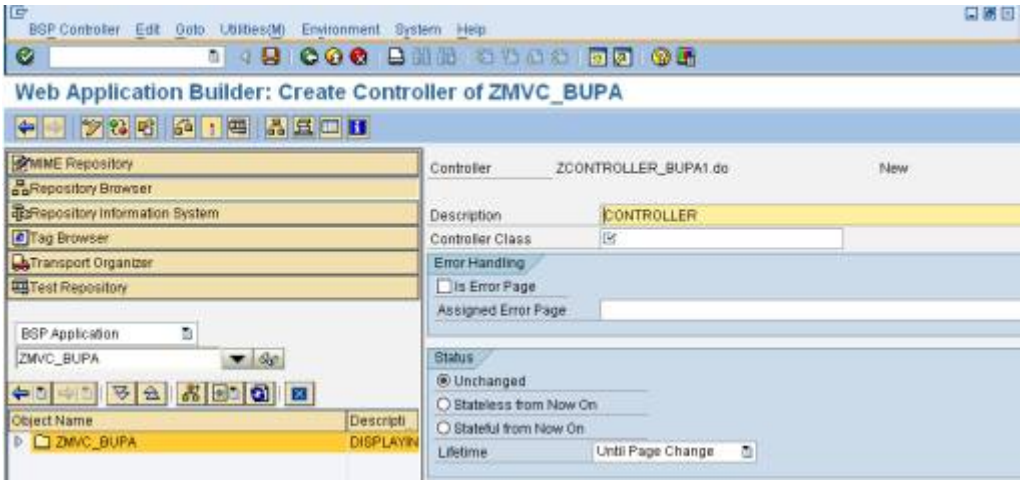
create a controller for the application as shown below. Right click on the application name → Create → Controller.



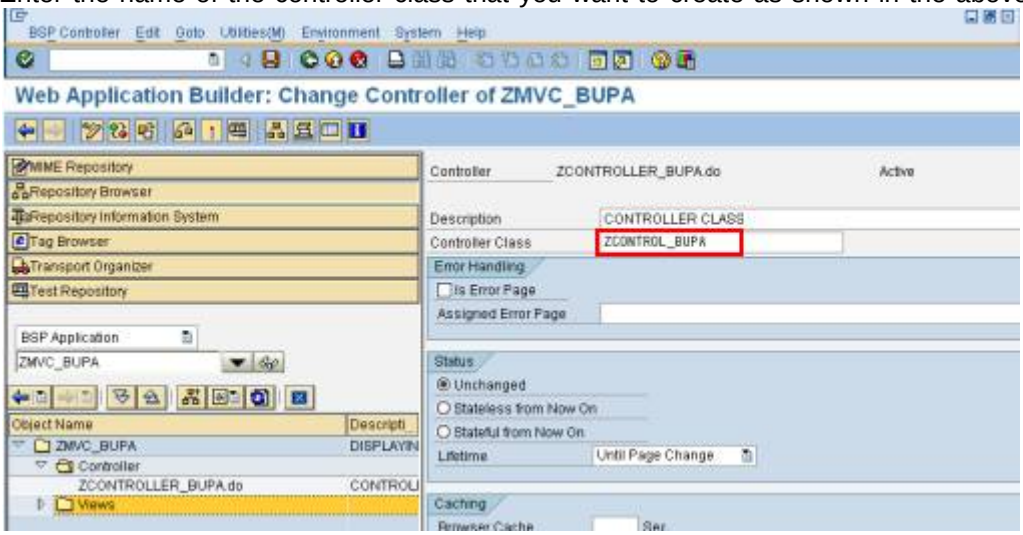
Input the description of the controller and click on the continue button.



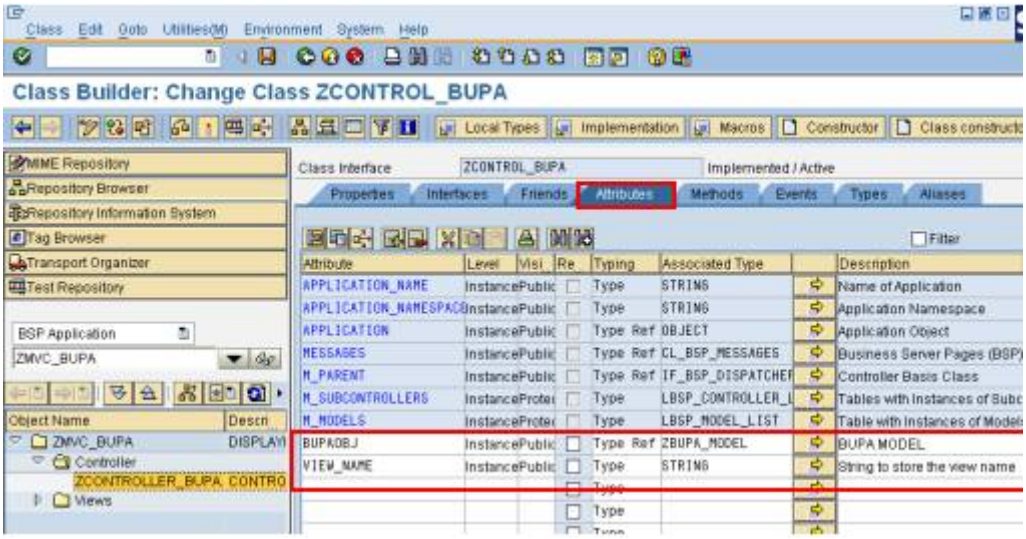
The screen appears as shown below.



Enter the name of the controller class that you want to create as shown in the above screen.



Double click on the controller class to enter into it. Give the attributes of the class as shown.



_ Redefining The Controller Class Methods

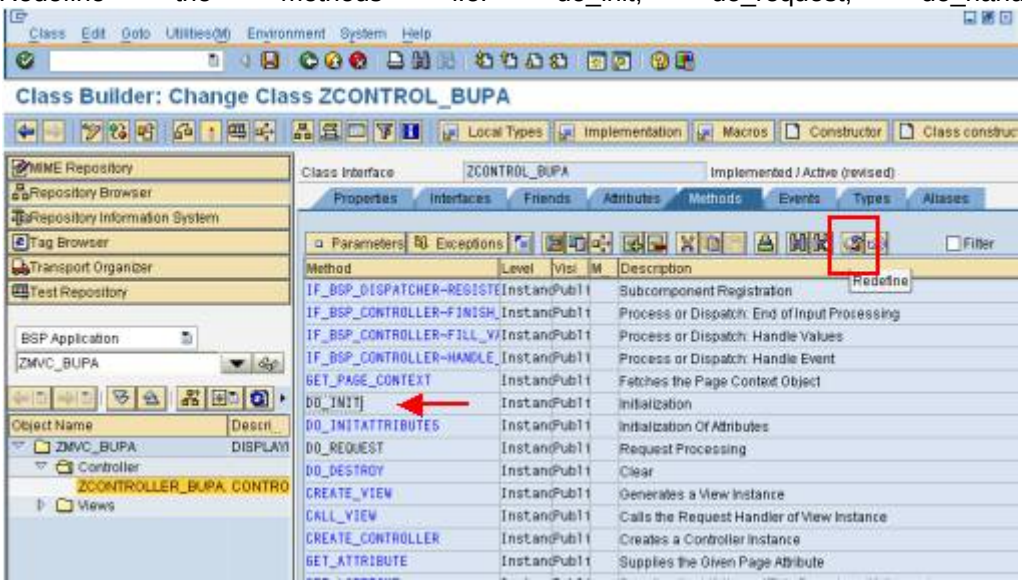
to use our model and views in controller class zcontroller_bupa.do, we have to redefine some of the methods inherited from base class cl_bsp_controller2 using the class builder's redefine function.

The redefinition steps automatically add commented coding (shown in bold in the do_init example below:

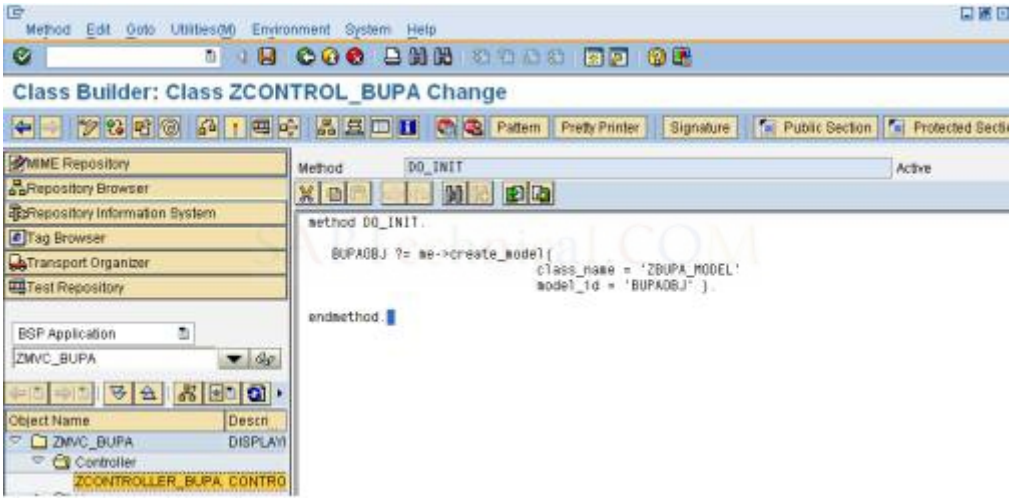
```
method do_init .
* call method super->do_init.
* .
endmethod.
```

If you uncomment those lines, you would execute the method implementation of the parent class. this is helpful when you want to carry out the parent logic and some additional custom logic. we don't want to reuse functionality of the parent class in the example, however, so be sure to delete the commented code before adding the new code.

Redefine the methods i.e. do_init, do_request, do_handle_event.



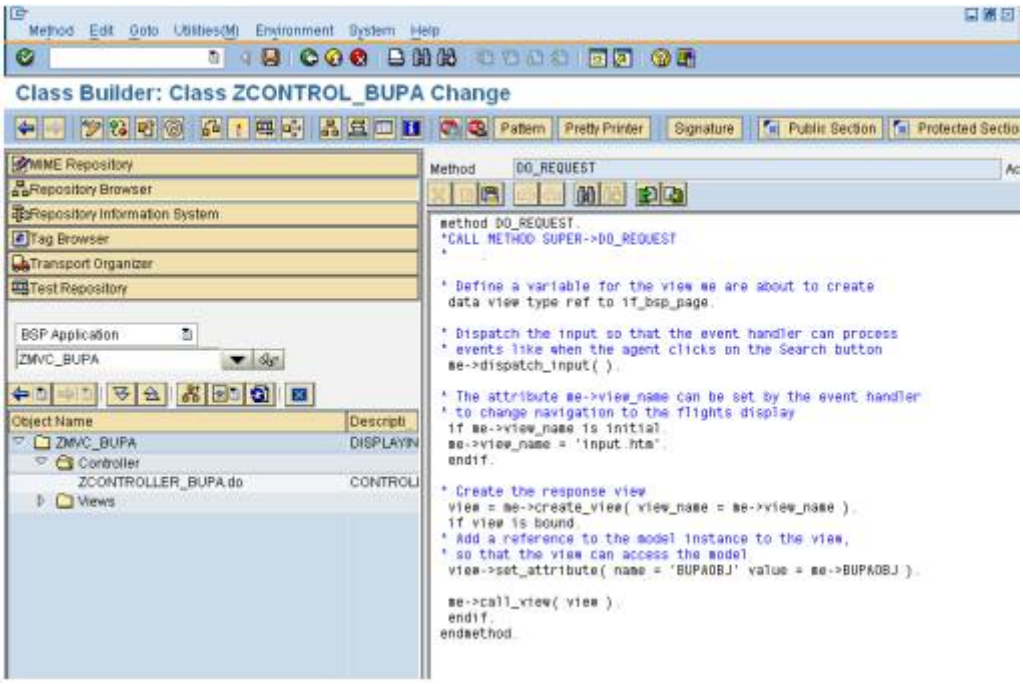
After that double click on the method do_init to implement it (write the code in the method).



CODE:

```
method DO_INIT.
   BUPAOBJ ?= me->create_model(
                              class_name = 'ZBUPA_MODEL'
                              model_id = 'BUPAOBJ' ).
endmethod.
```

Follows the same procedure for the ‘do_request’, ‘do_handle_event’.

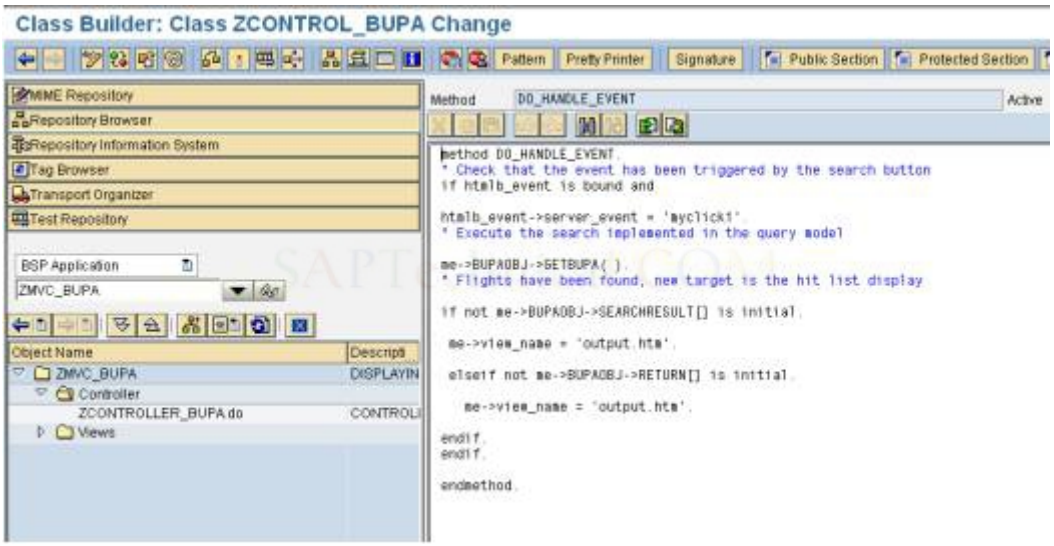


CODE:

```
method DO_REQUEST.
*CALL METHOD SUPER->DO_REQUEST
*    .
* Define a variable for the view we are about to create
 data view type ref to if_bsp_page.
* Dispatch the input so that the event handler can process
* events like when the agent clicks on the Search button
 me->dispatch_input( ).
* The attribute me->view_name can be set by the event handler
* to change navigation to the flights display
```

```
 if me->view_name is initial.
 me->view_name = 'input.htm'.
 endif.
* Create the response view
 view = me->create_view( view_name = me->view_name ).
 if view is bound.
* Add a reference to the model instance to the view,
* so that the view can access the model
 view->set_attribute( name = 'BUPAOBJ' value = me->BUPAOBJ ).
 me->call_view( view ).
 endif.
endmethod.
```

CLICK ON DO_HANDLE_EVENT METHOD.

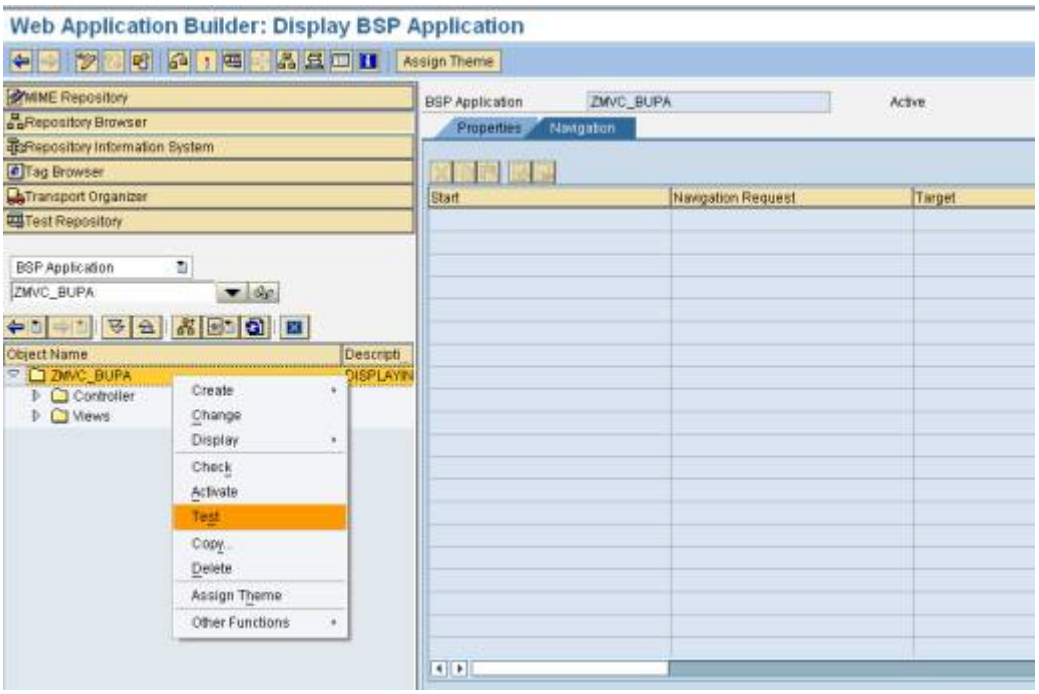


CODE:

```
method DO_HANDLE_EVENT.
* Check that the event has been triggered by the search button
if htmlb_event is bound and
htmlb_event->server_event = 'myclick1'.
* Execute the search implemented in the query model
me->BUPAOBJ->GETBUPA( ).
* Flights have been found, new target is the hit list display
if not me->BUPAOBJ->SEARCHRESULT[] is initial.
 me->view_name = 'output.htm'.
 elseif not me->BUPAOBJ->RETURN[] is initial.
   me->view_name = 'output.htm'.
endif.
endif.
endmethod.
```

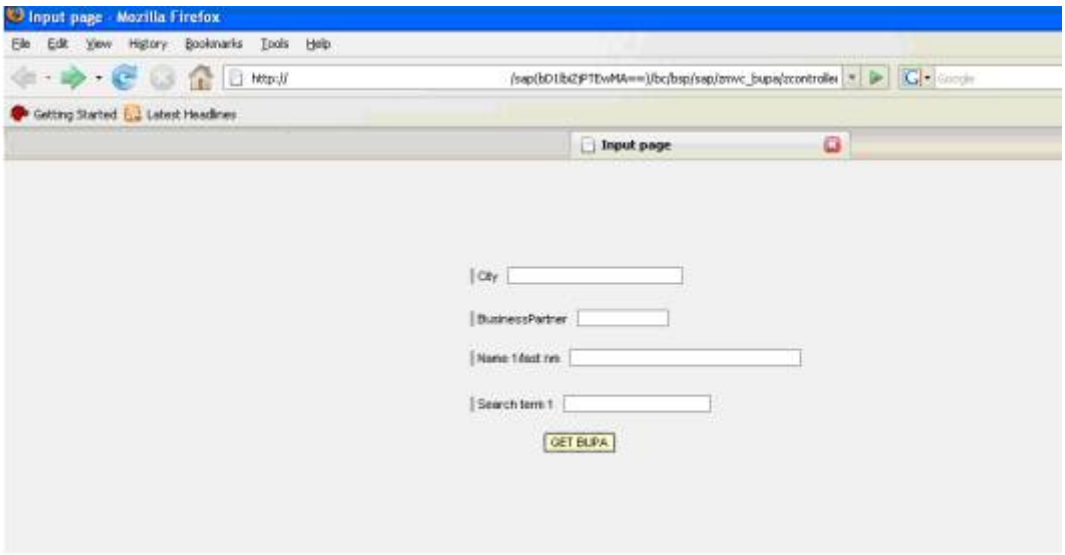
Save, check and activate the application.

Testing the application.

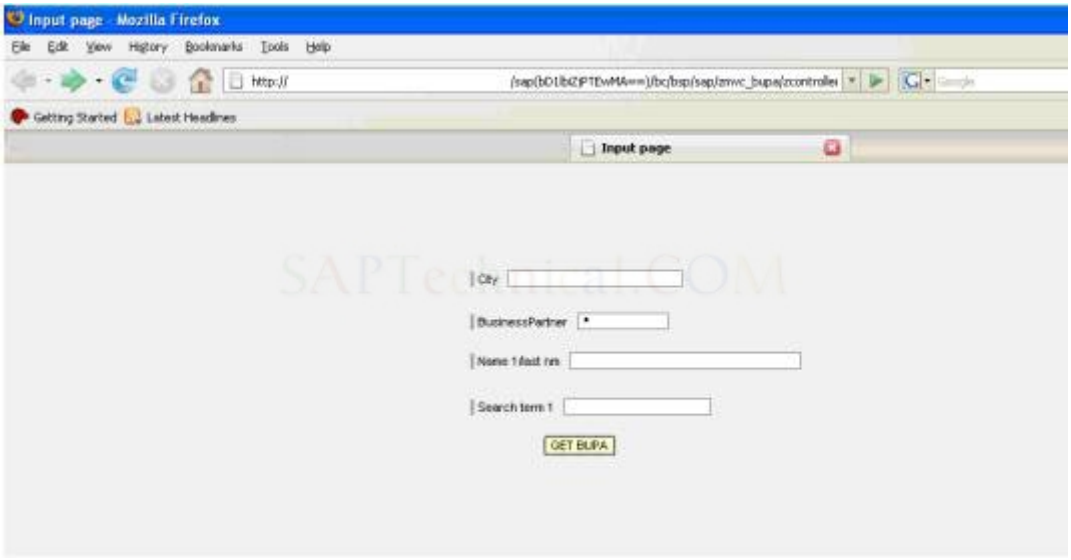


Input the user name and password for the authentication and click on ‘ok’ button.

The View is displayed in the browser window as shown below.



Provide the information to get business partner data.



Click on the GET DATA button.

Then business partner information gets displayed as follows