AdSaDi - Advanced SAP Dialog

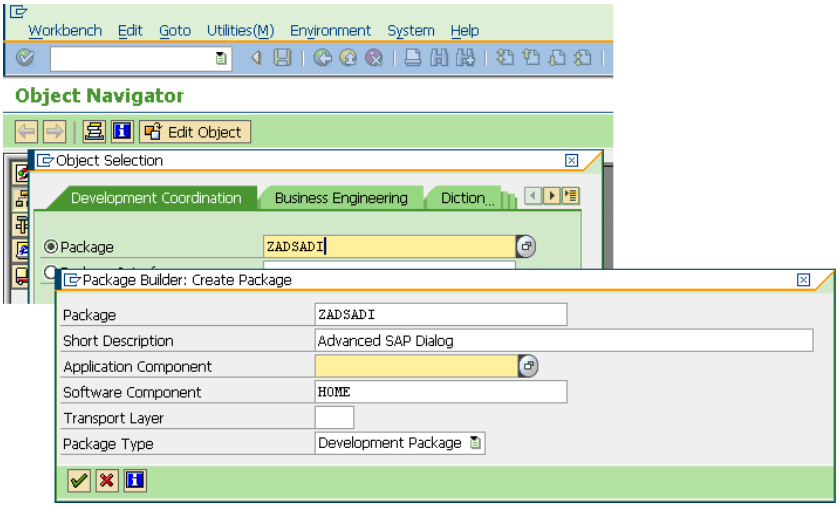
Author: Frithjof Eckhardt, updated: 25. Okt. 2013

Installation

Note: The package comes with the namespace /UKW/, which is the namespace for Wuerzburg University Hospital, Germany. You can't create objects in this namespace in your SAP system. Therefore map the namespace prefix of each object from /UKW/ to any name starting with Z or Y, or your own customer namespace.

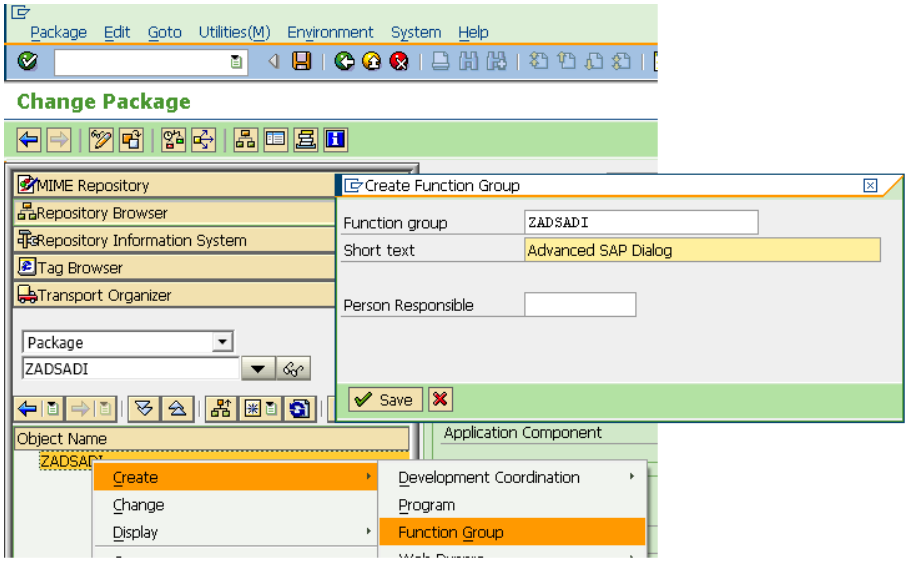
1. Package ZADSADI
2. Function Group ZADSADI
3. Function Module Z_ADSADI with one screen 100 carrying a custom control
4. Create dictionary objects for storing html content
5. Interface ZIF_ADSADI_CALLBACK
6. Class ZCL_ADSADI_DIALOG

Create new Package, e.g. ZADSADI

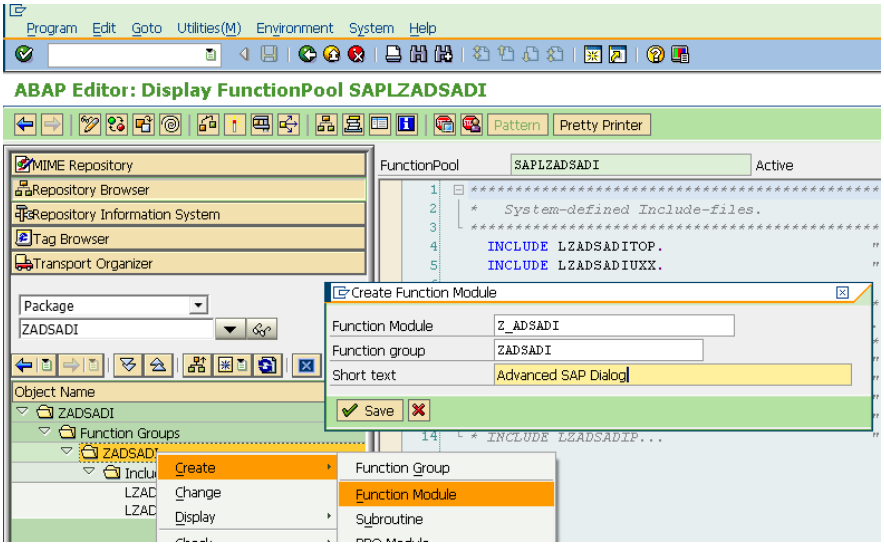


AdSaDi - Advanced SAP Dialog

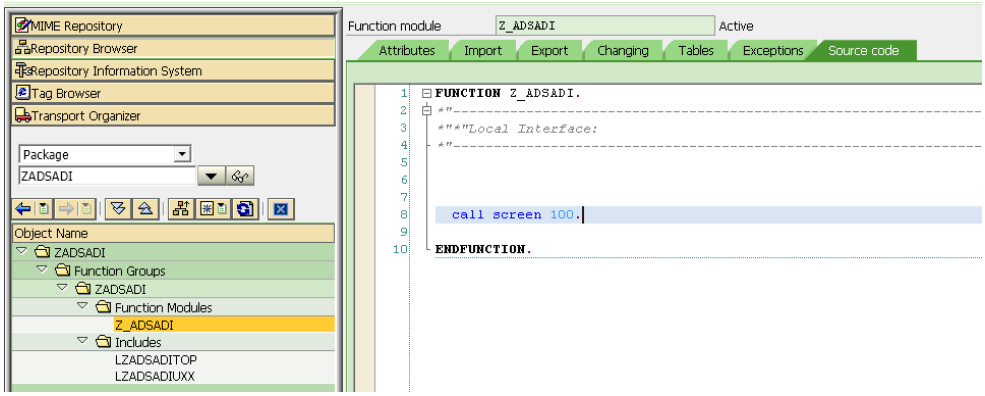
Within the package create and activate Function Group, e.g. ZADSADI



Within the function group create Function Module, e.g. Z_ADSADI

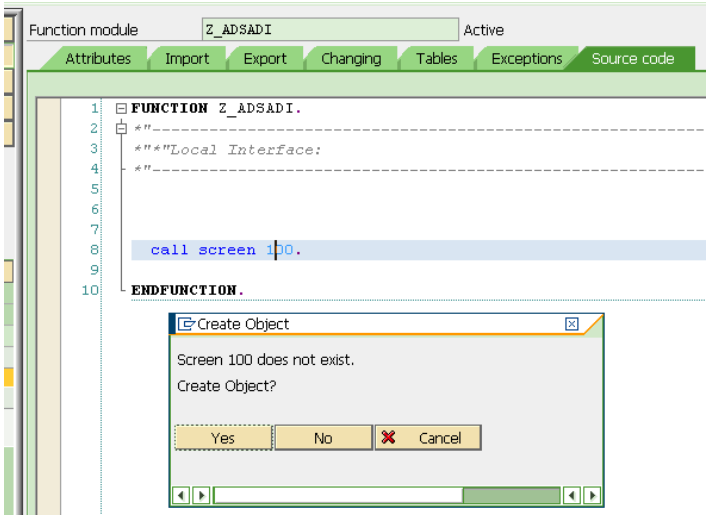


Add the following line to the source code of the function module and activate

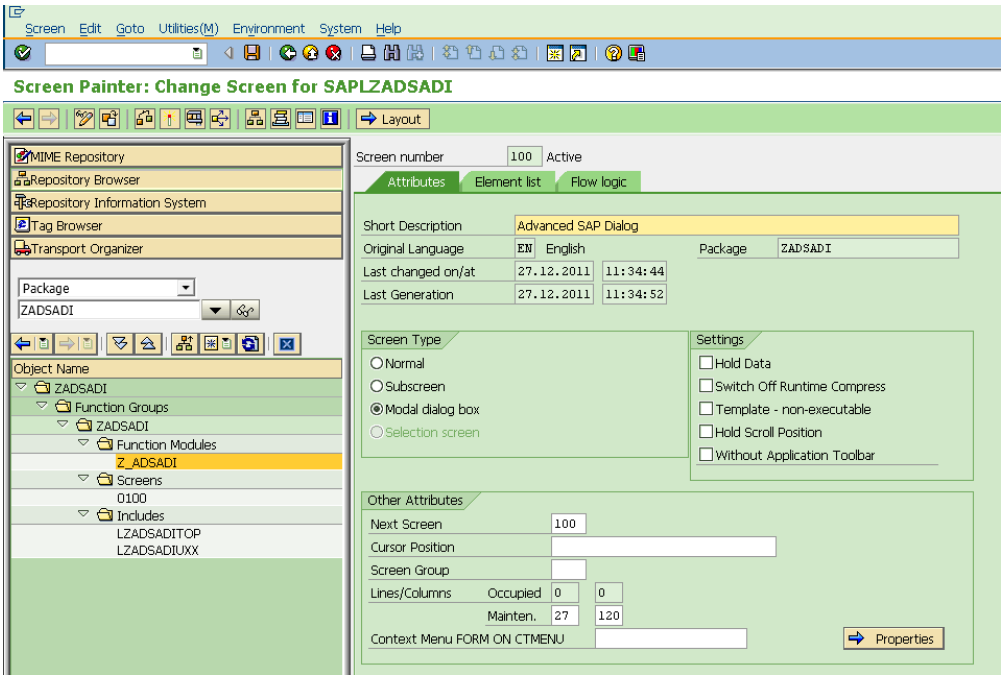


AdSaDi - Advanced SAP Dialog

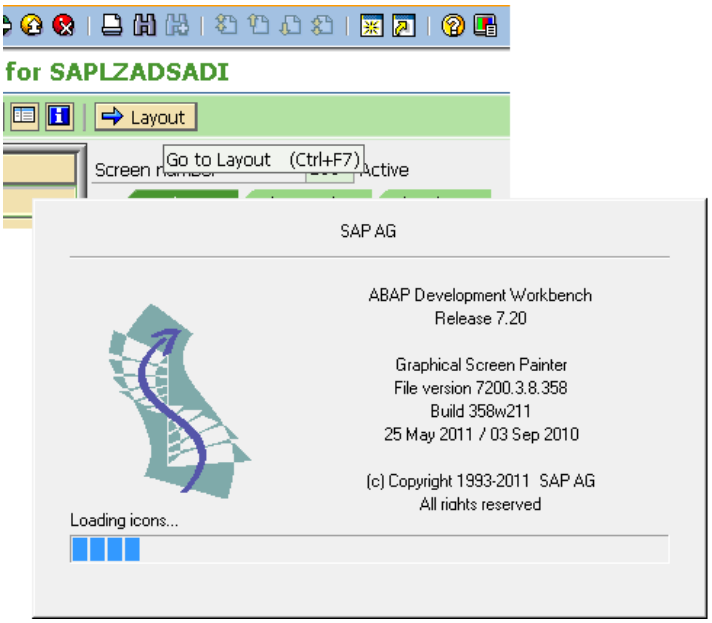
Create screen 100 by double clicking the „100“ in the source code:



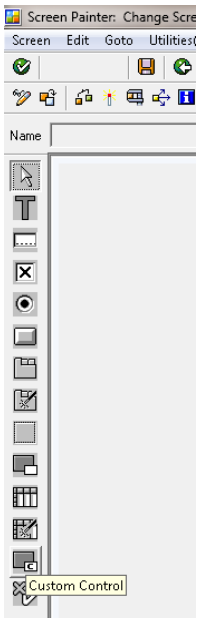
Set the screen type to „Modal dialog box“ and activate



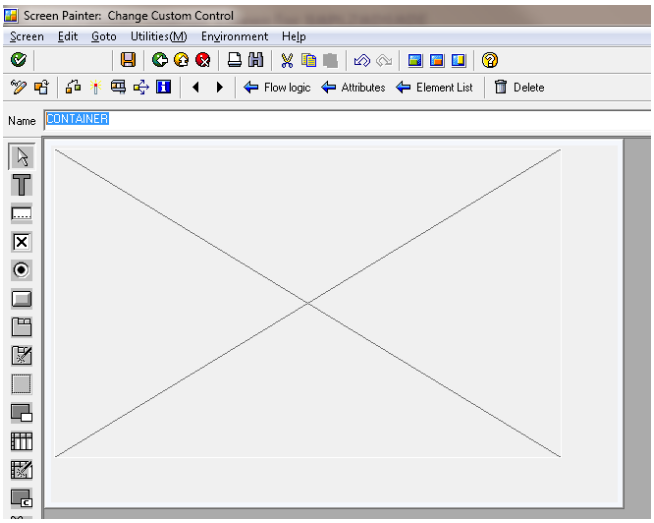
Create the layout for the screen



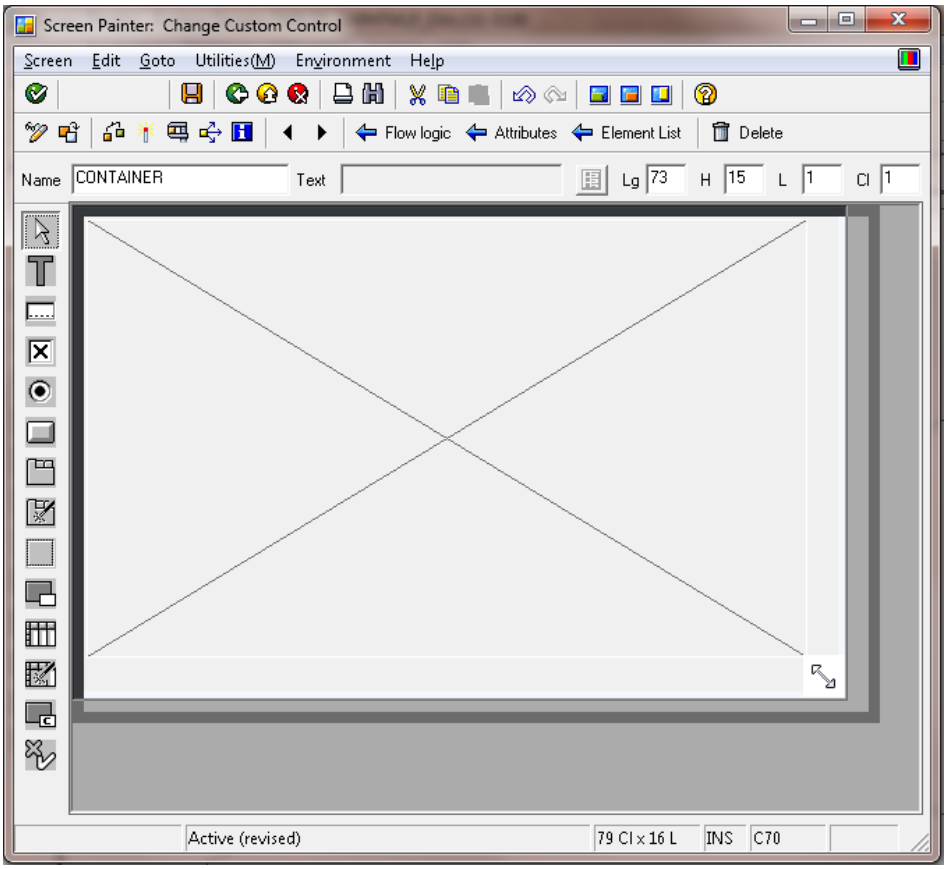
Create a custom control screen element



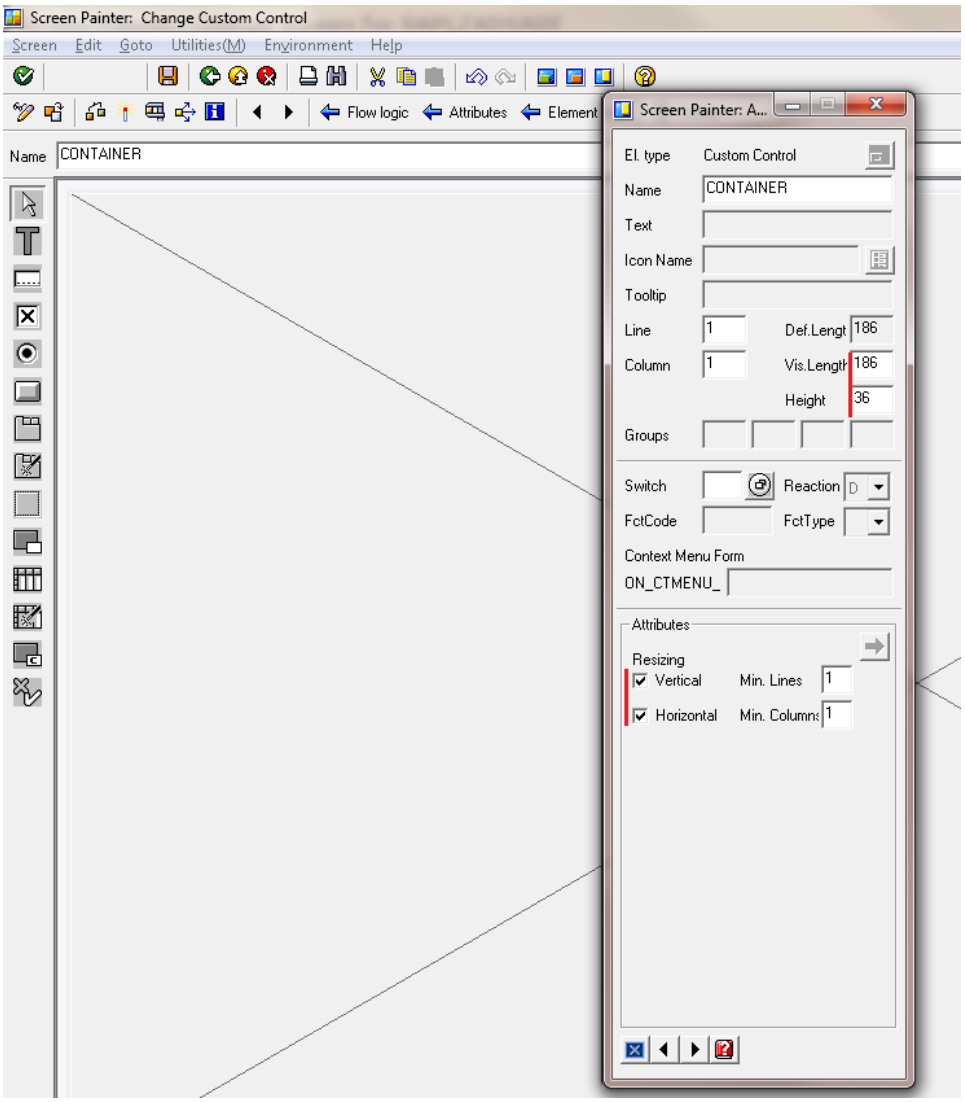
Set the name of the control to „CONTAINER“

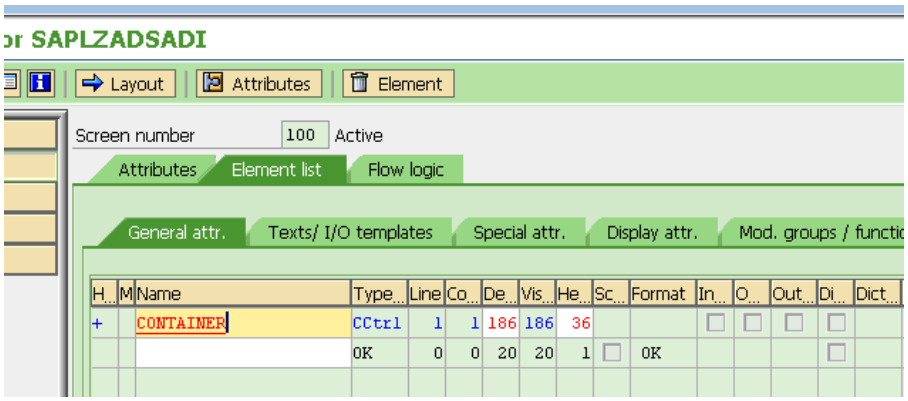


Resize the window below the container so that it is a little greater than the screen size of the target monitor , e.g. 186 columns, 36 rows for 19“ monitor (trial&error determined)



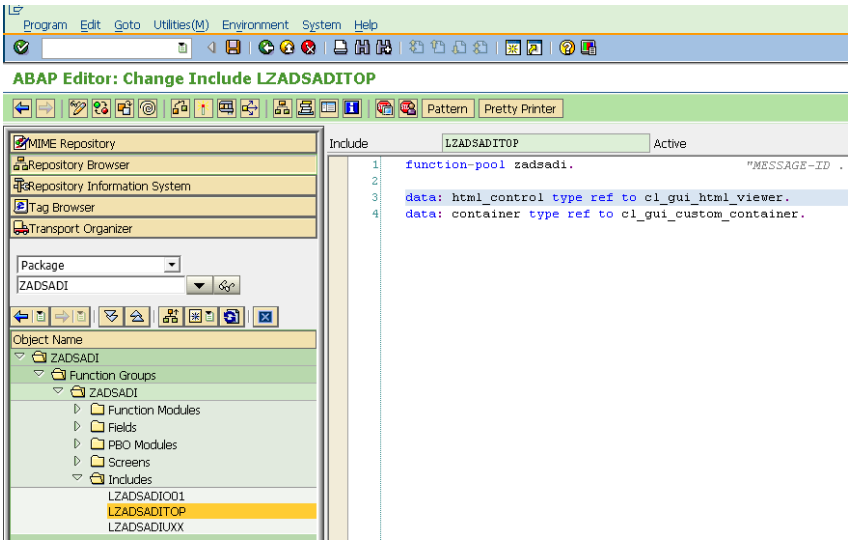
Set the dimension of the container to the same size as the underlying window and link both together by setting the resizing attributes of the container and activate



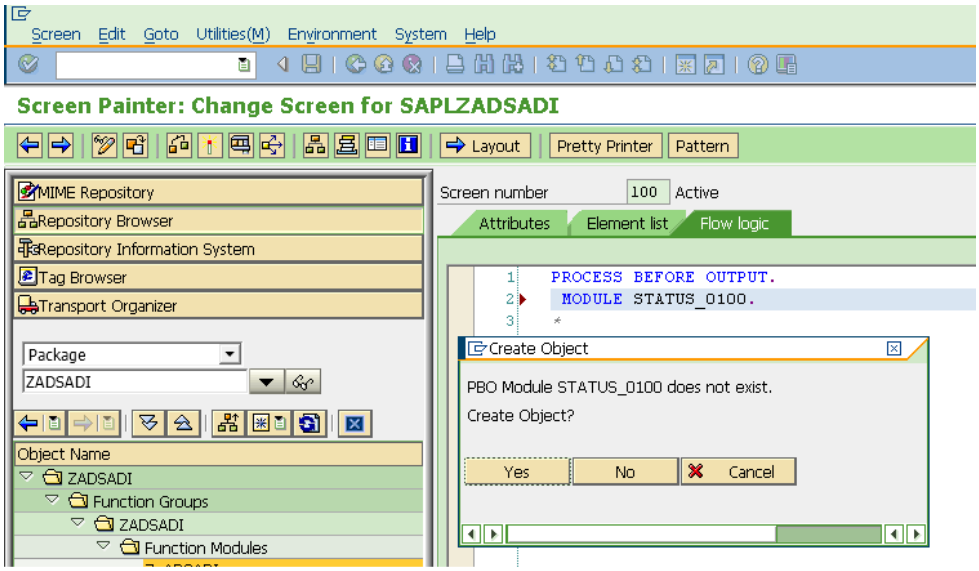


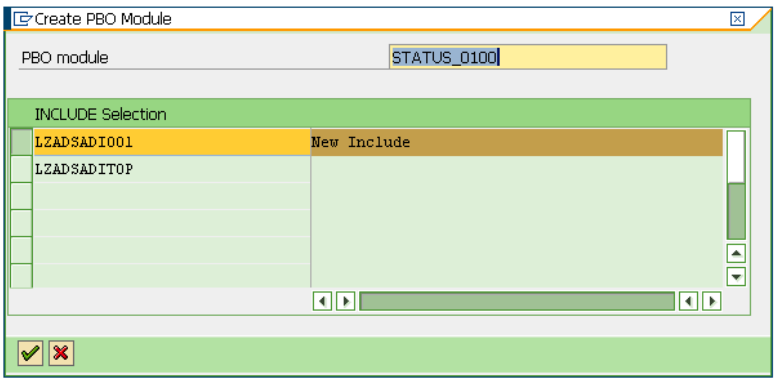
Edit the top include of the function module and enter the following data definitions

```
data: html_control type ref to cl_gui_html_viewer.
data: container type ref to cl_gui_custom_container.
```



Create Flow logic of the screen for „PROCESS BEFORE OUTPUT“ by uncommenting the respective module call and creating the source code by clicking twice on the module name „STATUS_0100“

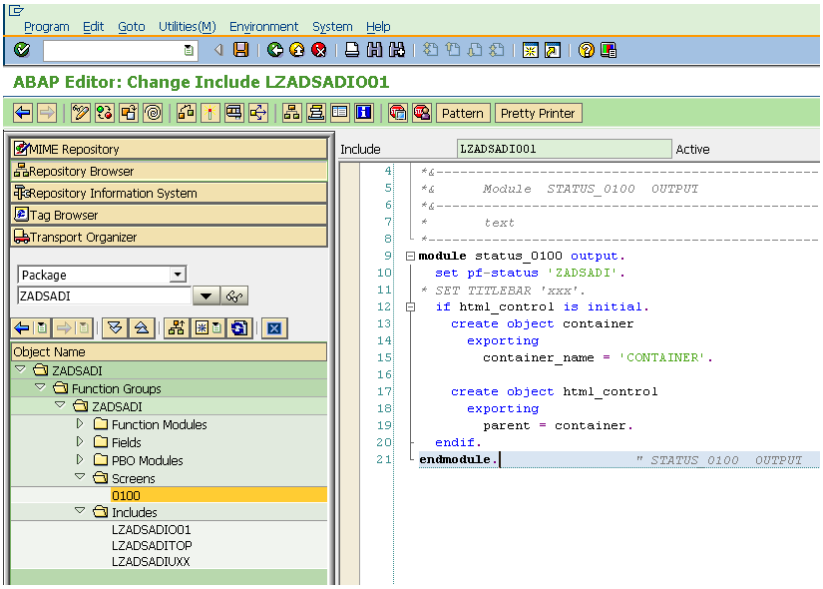




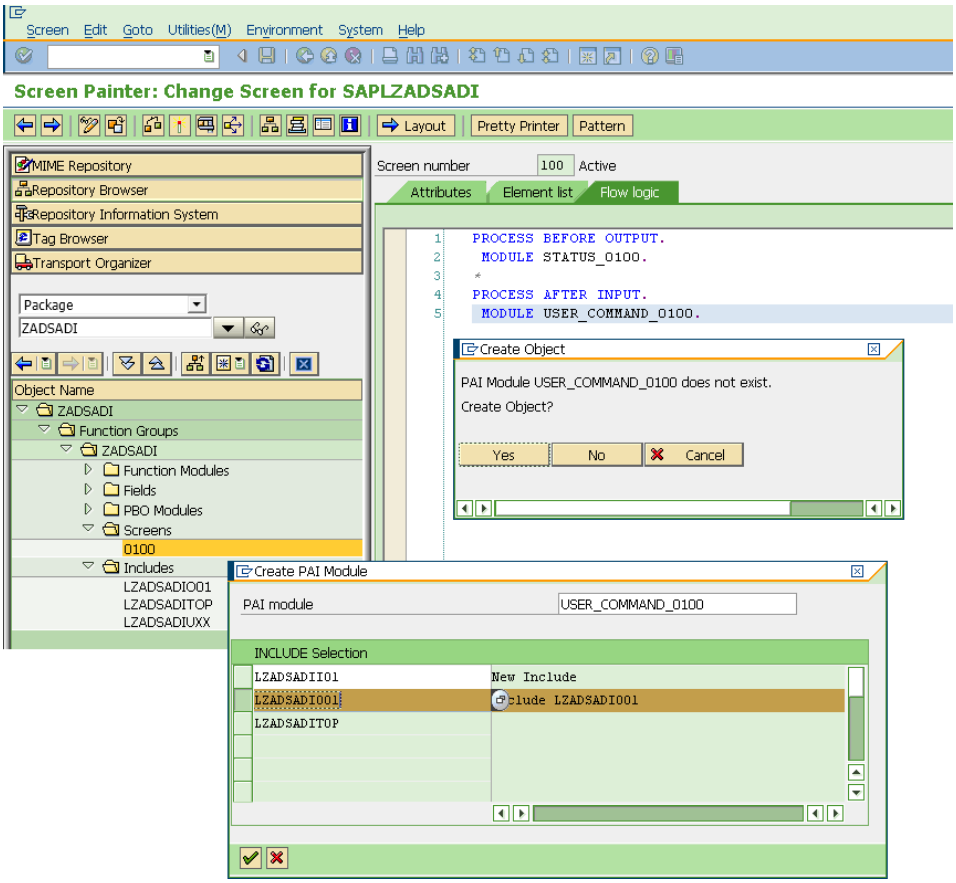
Enter the following code

```
module status_0100 output.
  set pf-status 'ZADSADI'.
```

```
* SET TITLEBAR 'xxx'.
  if html_control is initial.
    create object container
      exporting
        container_name = 'CONTAINER'.

    create object html_control
      exporting
        parent = container.
  endif.
endmodule.
```



In the same include, create Flow logic of the screen for „PROCESS AFTER INPUT“ by uncommenting the respective module call and creating the source code by clicking twice on the module name „USER_COMMAND_0100“

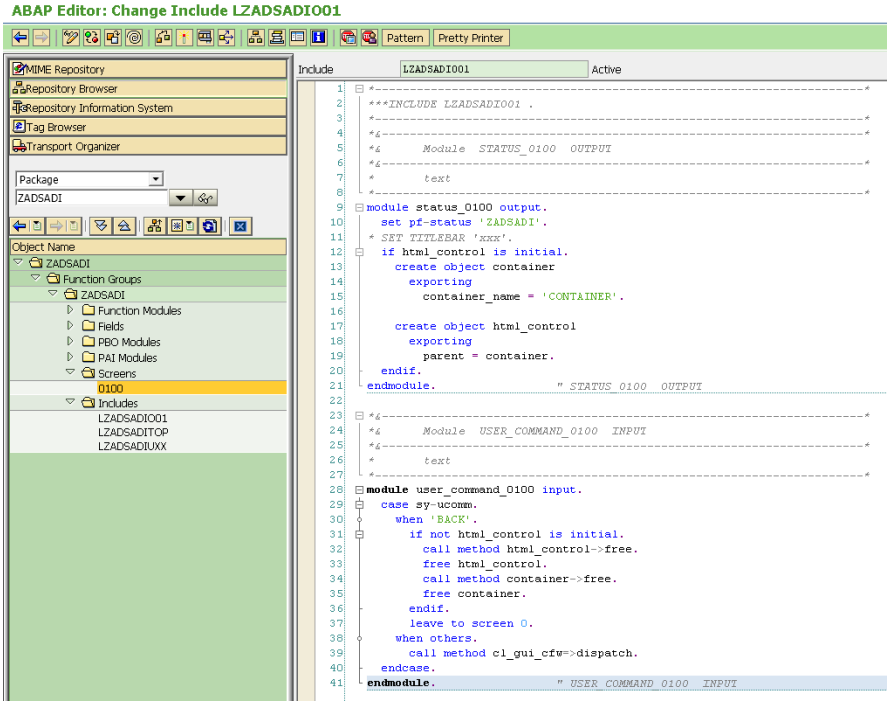


Enter the following code

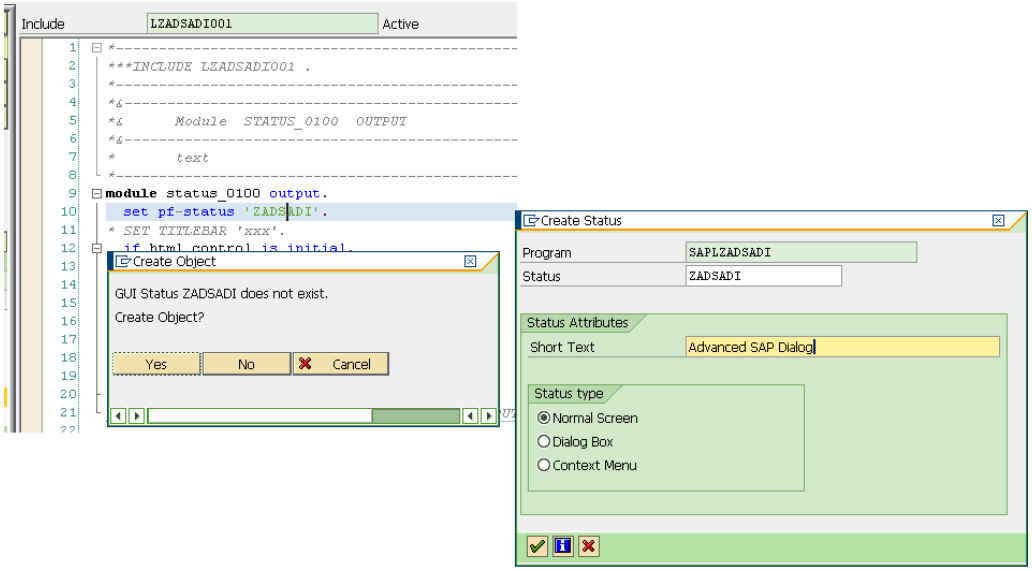
```
module user_command_0100 input.
  case sy-ucomm.
    when 'BACK'.
      if not html_control is initial.
        call method html_control->free.
        free html_control.
        call method container->free.
        free container.
      endif.
      leave to screen 0.
    when others.
      call method cl_gui_cfw=>dispatch.
  endcase.
endmodule.
```

AdSaDi - Advanced SAP Dialog

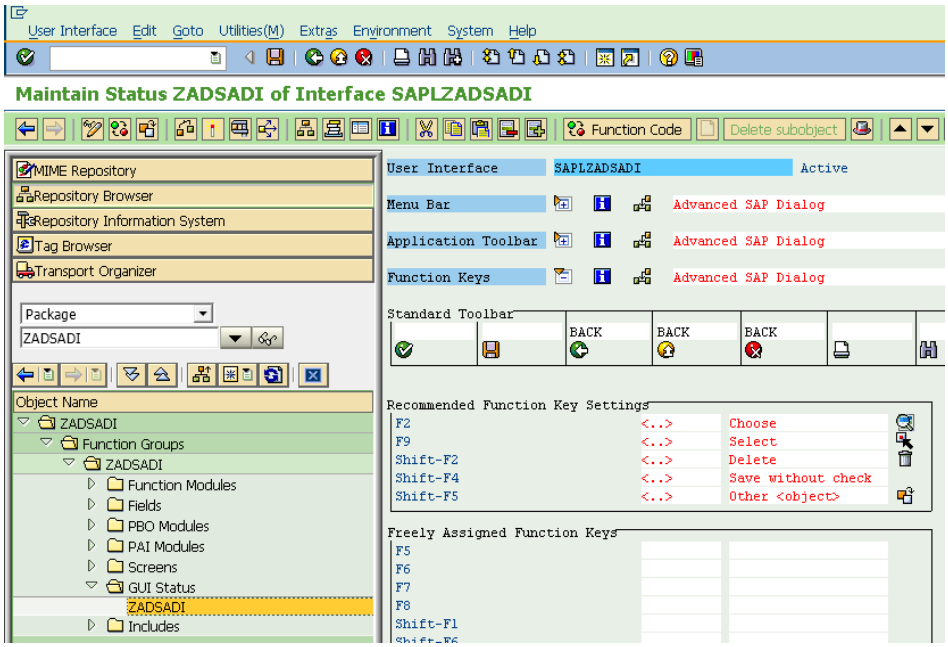
ABAP Editor: Change Include LZADSADIO01



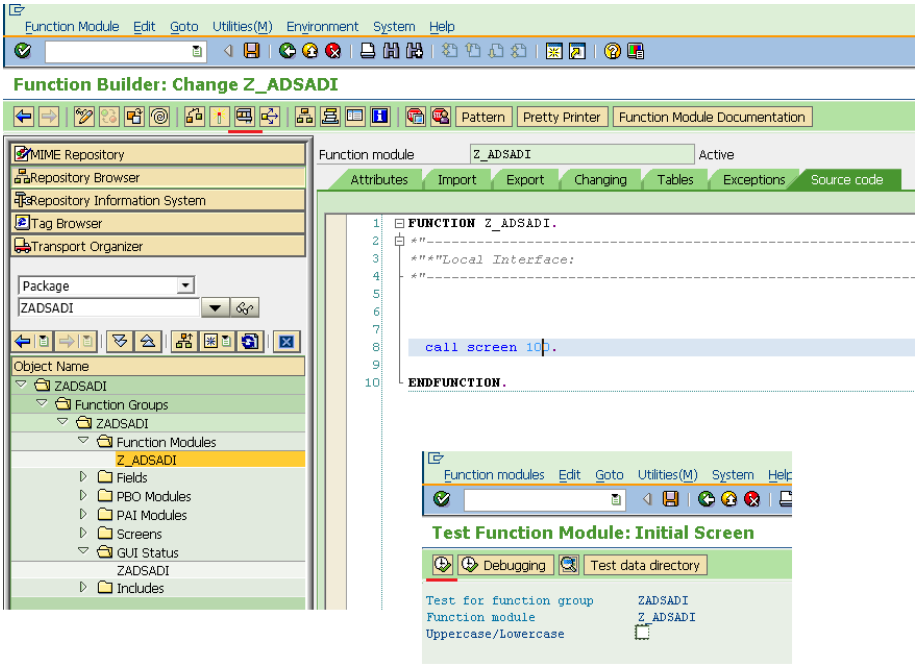
Create the GUI status by double-clicking on the „ZADSADI“ in the status_0100 module in the line „set pf-status 'ZADSADI'



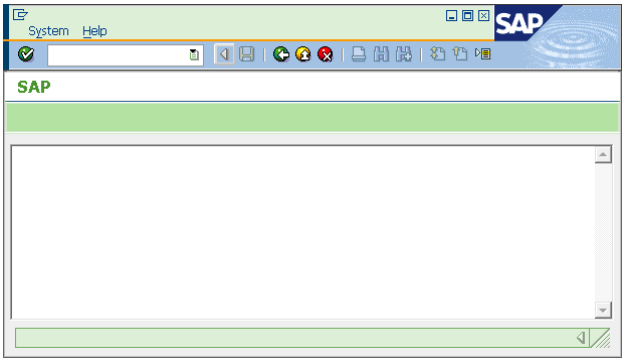
Enter the function code „BACK“ for all the navigation buttons (back, exit, cancel) in the toolbar section and activate the status



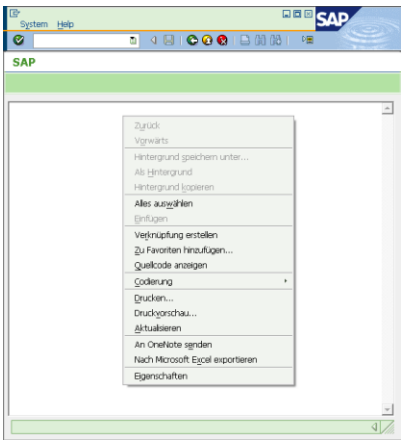
Test the screen by testing the function module



A white window appears, that is resizeable and any navigation button brings the user back



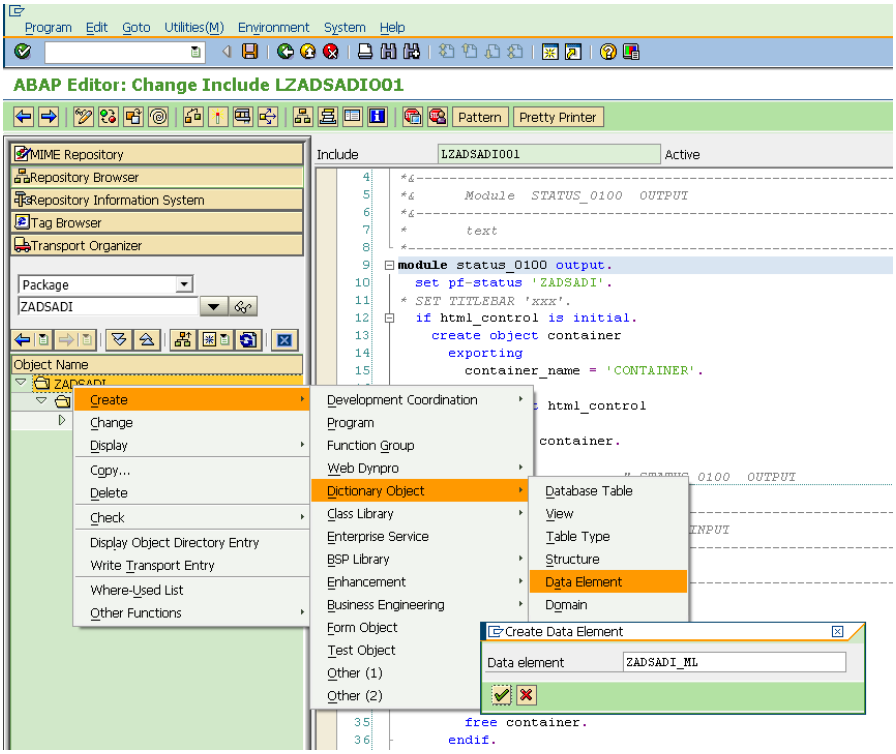
The white window is an html page, verify it by hiting the right mouse button to show the Internet Explorer context menu

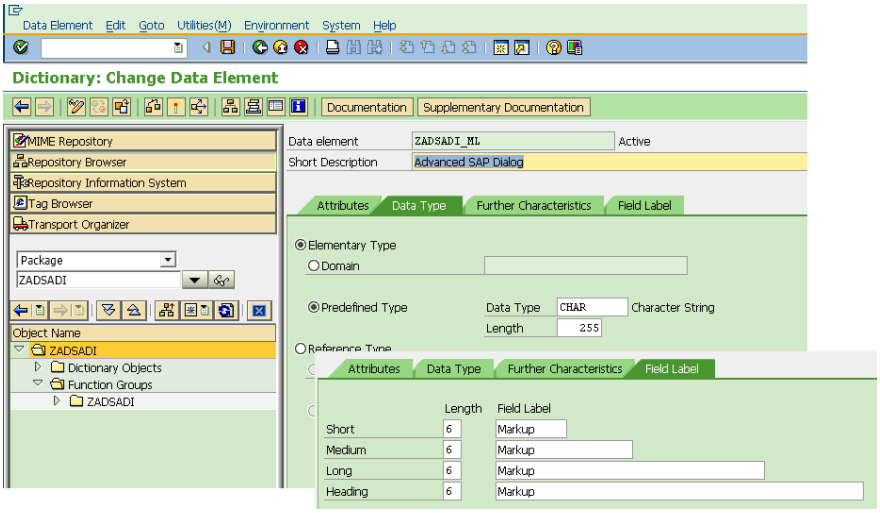


Create the backing logic

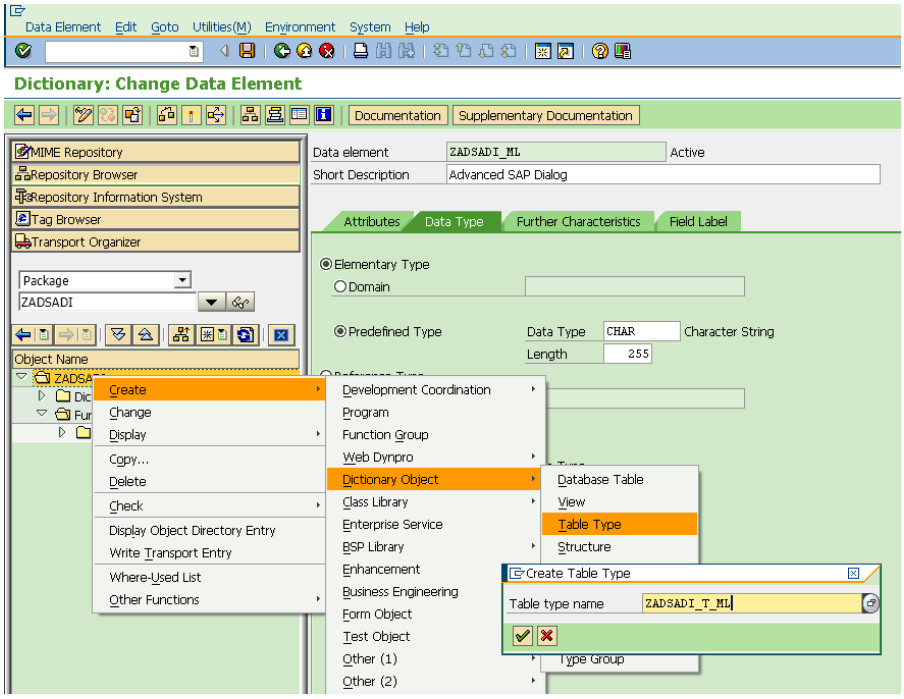
AdSaDi - Advanced SAP Dialog

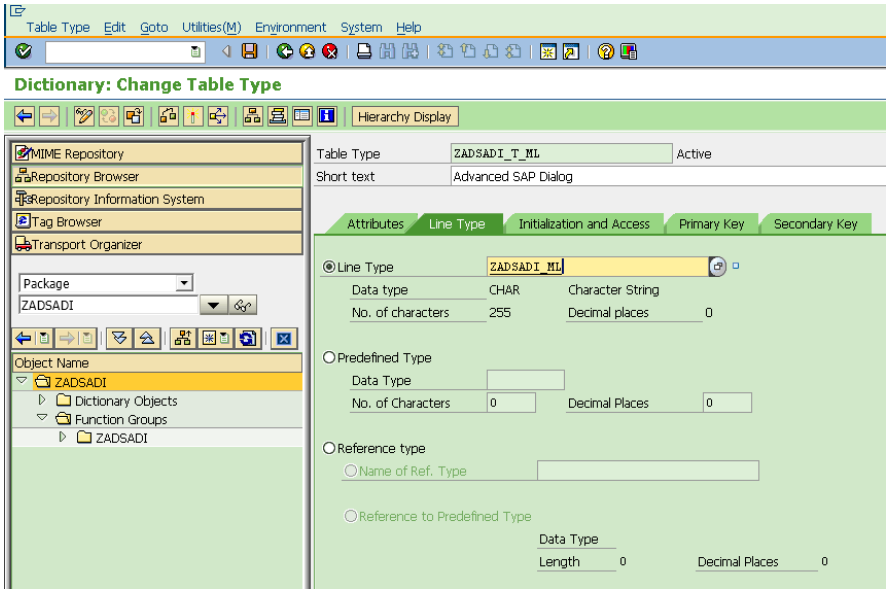
Create a data element ZADSADI_ML for storing a line of html markup





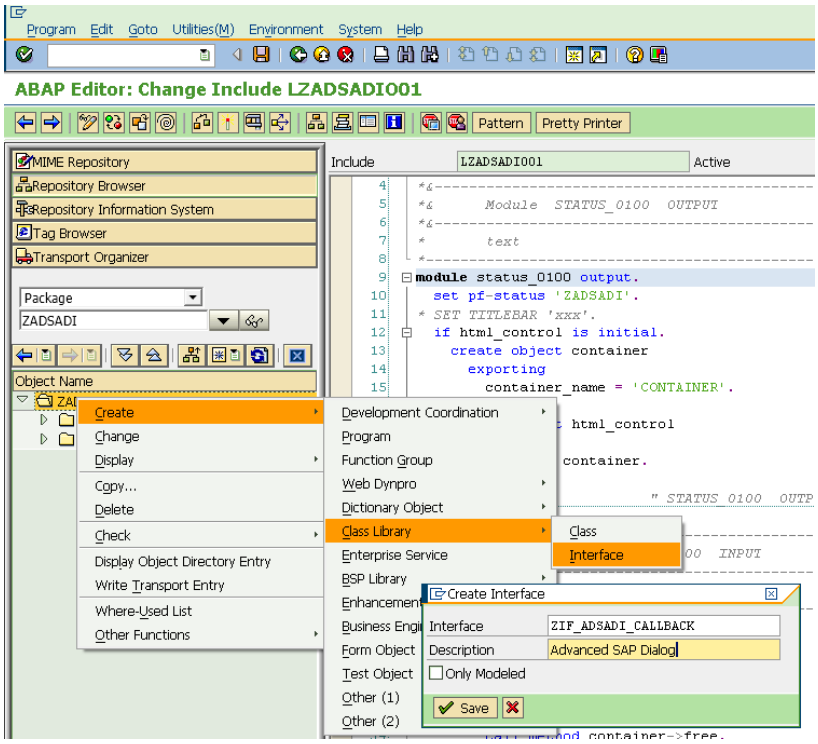
Create a table type ZADSADI_T_ML for html content

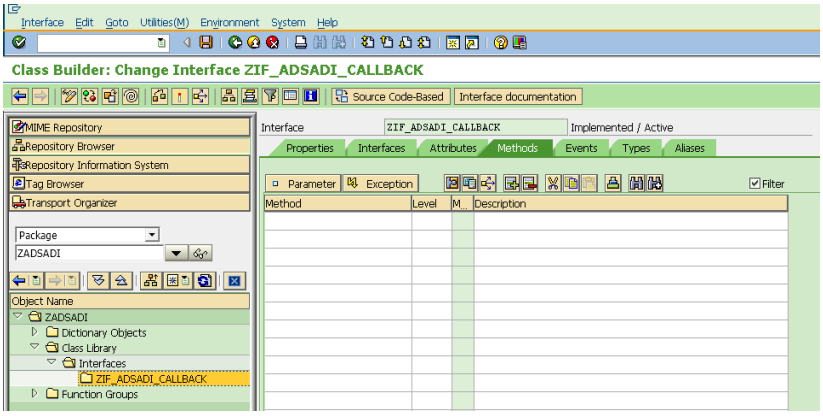




Create an interface ZIF_ADSADI_CALLBACK

This interface is used from within the screen 100 implementation to get the html content to be displayed and to provide the parameters returned from the html viewer.



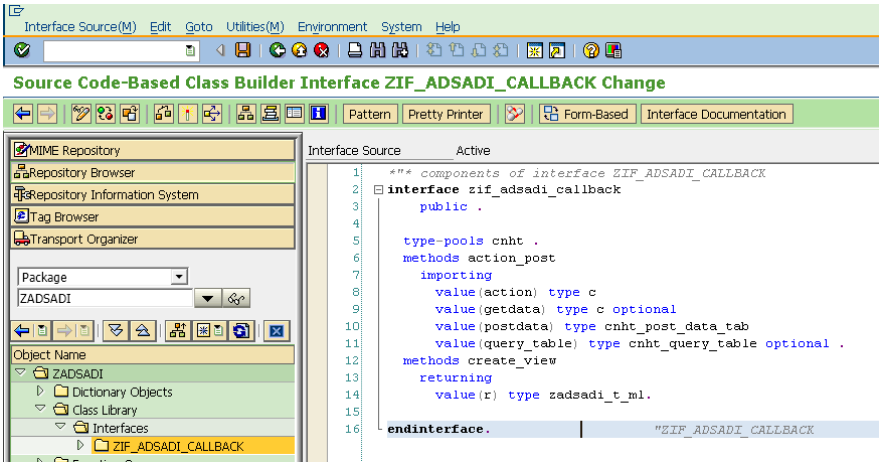


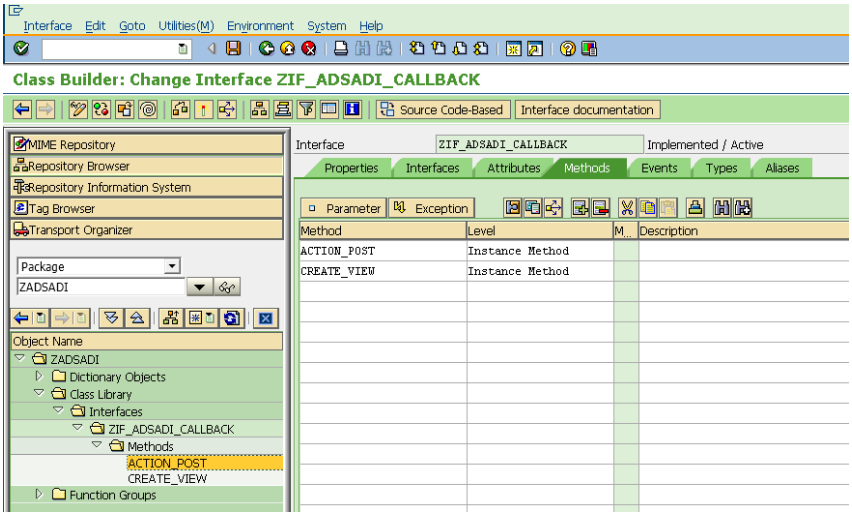
Switch to „Source Code-Based“ view of the class builder. If this button is not available (due to an older release), create the methods the traditional way one at a time with the class builders help.

In the source code-based view, enter the following interface code and switch back to „Form-based“ view.

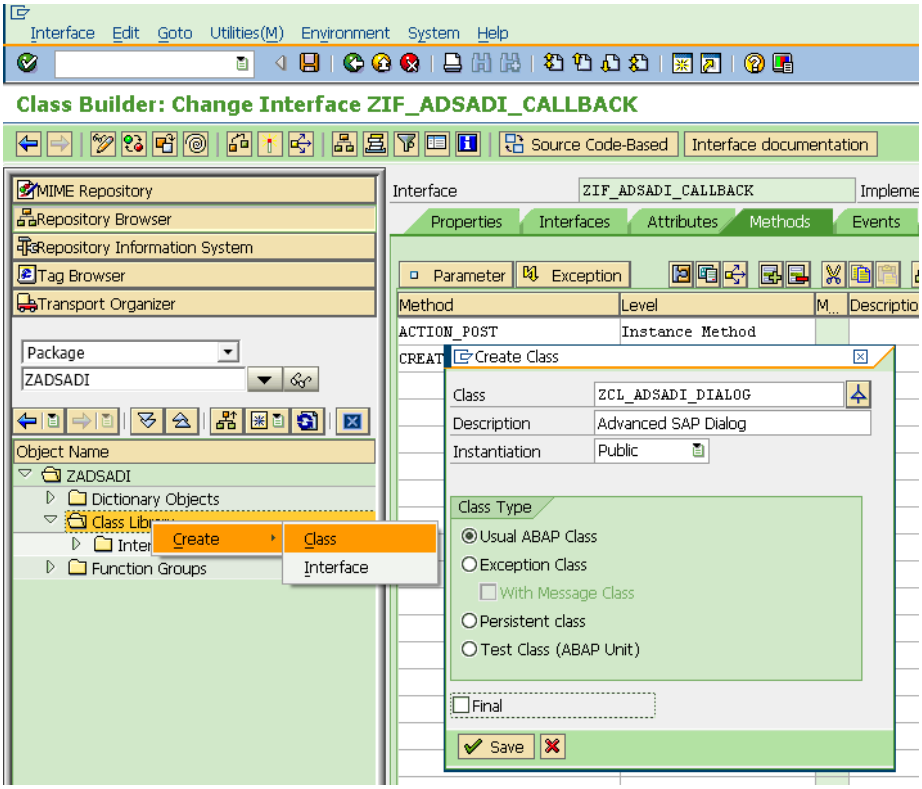
```
interface zif_adsadi_callback public.

  type-pools cnht.

  constants dlg_cancel type i value 1.
  constants dlg_no_cancel type i value 0.

  methods action_post
    importing
      value(action) type c
      value(getdata) type c optional
      value(postdata) type cnht_post_data_tab
      value(query_table) type cnht_query_table optional .
```

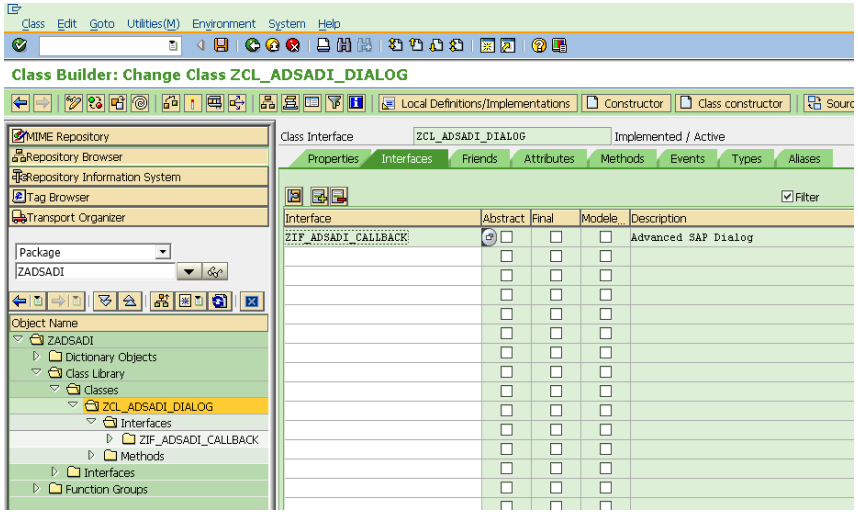
```
  methods create_view
    returning
      value(r) type zadsadi_t_ml.

endinterface.
```



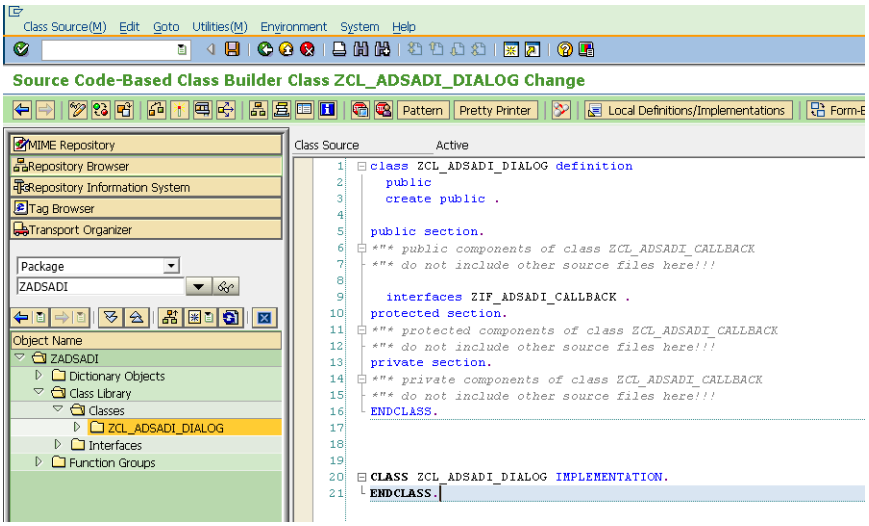


Create an implementation ZCL_ADSADI_DIALOG against this interface





Switch to „Source Code-based“ view and edit the source code the following way:



Public section

```
interfaces zif_adsadi_callback .

aliases create_view
  for zif_adsadi_callback~create_view.
aliases do_post
  for zif_adsadi_callback~action_post.

type-pools abap.
type-pools cnht.

methods constructor
  importing
    value(i_xslt) type csequence
    value(i_xml) type csequence optional
    value(i_xml_doc) type ref to if_ixml_node optional
    value(i_params) type abap_trans_parmbind_tab optional
    value(i_size) type csequence default ''
    value(i_top_left_y) type i default 3
    value(i_top_left_x) type i default 40
    value(i_width) type i default 100
    value(i_height) type i default 25.
methods get_value
  importing
    value(n) type csequence
  returning
    value(r) type string .
methods render
  returning
    value(canceled) type i .
```

Protected section

```
data xml type string .
data xml_doc type ref to if_ixml_node .
```

Private section

```
    data xslt type string.
    data xslt_params type abap_trans_parmbind_tab.
    data query_table type cnht_query_table.
    data size type string.
    data top_left_y type i.
    data top_left_x type i.
    data width type i.
    data height type i.
```

Constructor **implementation**

```
  method constructor.
    me->xslt_params  = i_params.
    me->xslt         = i_xslt.
    me->size         = i_size.
    me->top_left_y   = i_top_left_y.
    me->top_left_x   = i_top_left_x.
    me->width        = i_width.
    me->height       = i_height.
    if i_xml is not initial.
      me->xml = i_xml.
    elseif i_xml_doc is bound.
      me->xml_doc = i_xml_doc.
    else.
      concatenate
        `<?xml version="1.0" encoding="utf-8" ?>`
        `<dialog />`
        into me->xml.
    endif.
  endmethod.
```

Method get_value **implementation**

```
  method get_value.
    data: the_param type w3query.

    loop at me->query_table into the_param where name eq n.
      r = the_param-value.
      exit.
    endloop.
  endmethod.
```

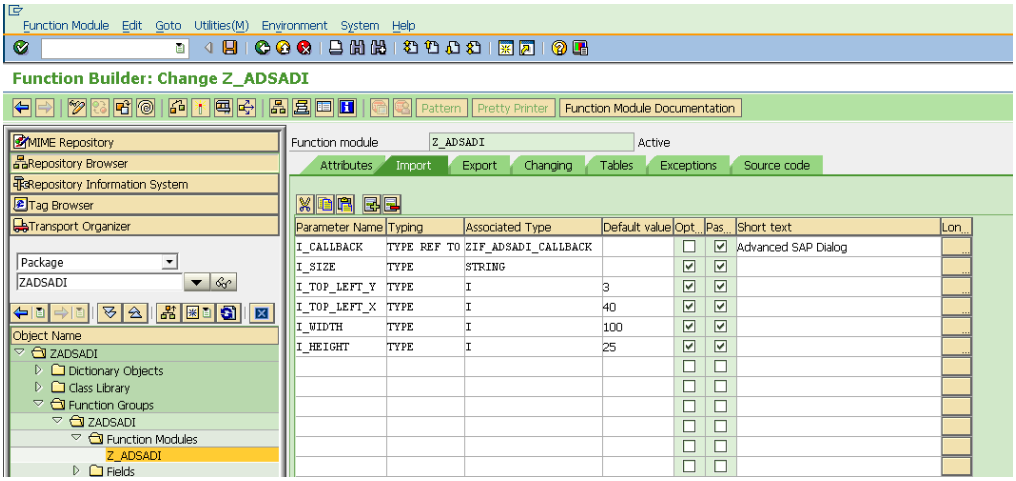
Method render **implementation**

```
  method render.
    call function 'Z_ADSADI'
      exporting
        i_callback   = me
        i_size       = me->size
        i_top_left_y = me->top_left_y
        i_top_left_x = me->top_left_x
        i_width      = me->width
        i_height     = me->height
```

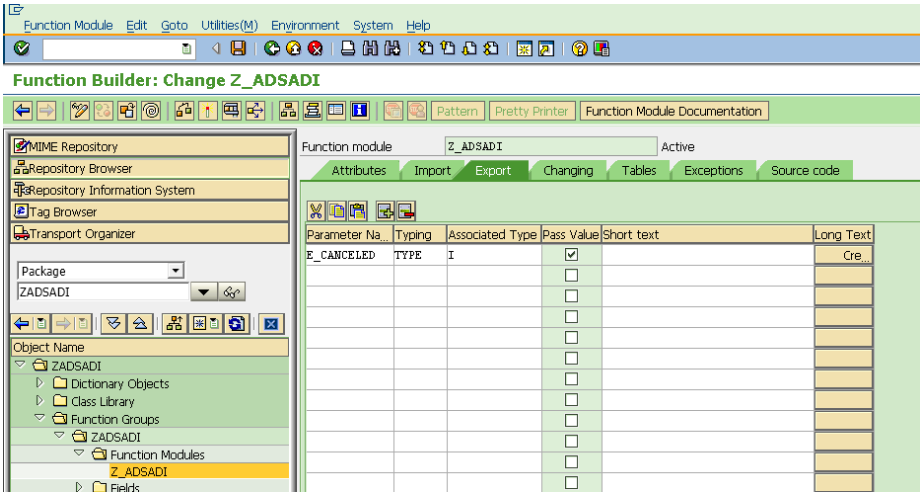
```
    importing
      e_canceled    = canceled.
endmethod.
```

Modify the interface of the function module to match the call from the render Method

Import



Export



Source code function module Z_ADSADI

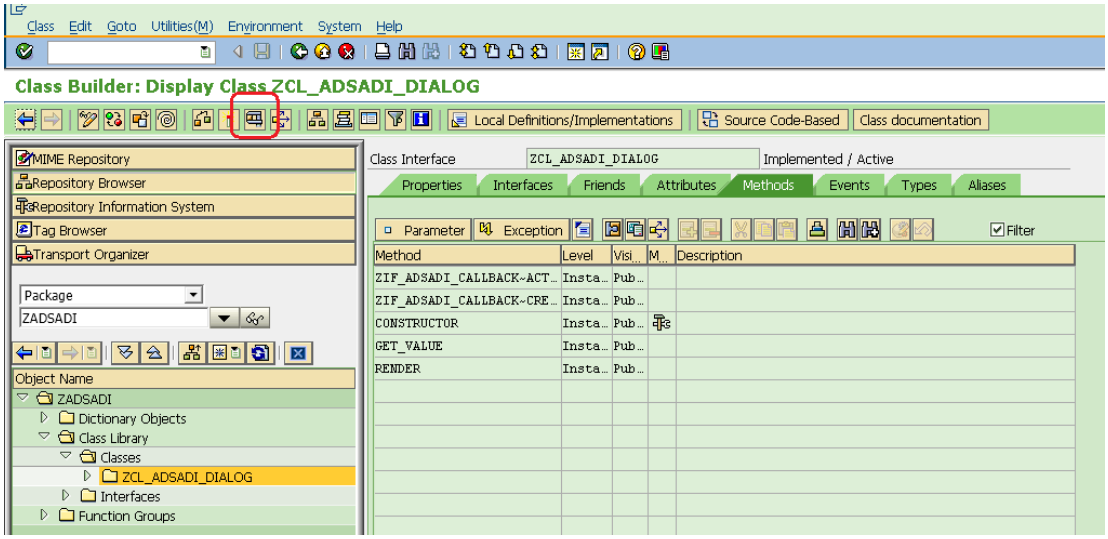
```
g_callback = i_callback.
g_canceled = zif_adsadi_callback=>dlg_cancel.

if i_size eq 'NORMAL'.
  call screen 100
    starting at 40  3
    ending   at 140 30.
elseif i_size eq 'FULL'.
  call screen 100.
else.
  data: bottom_right_x type i.
  data: bottom_right_y type i.
```

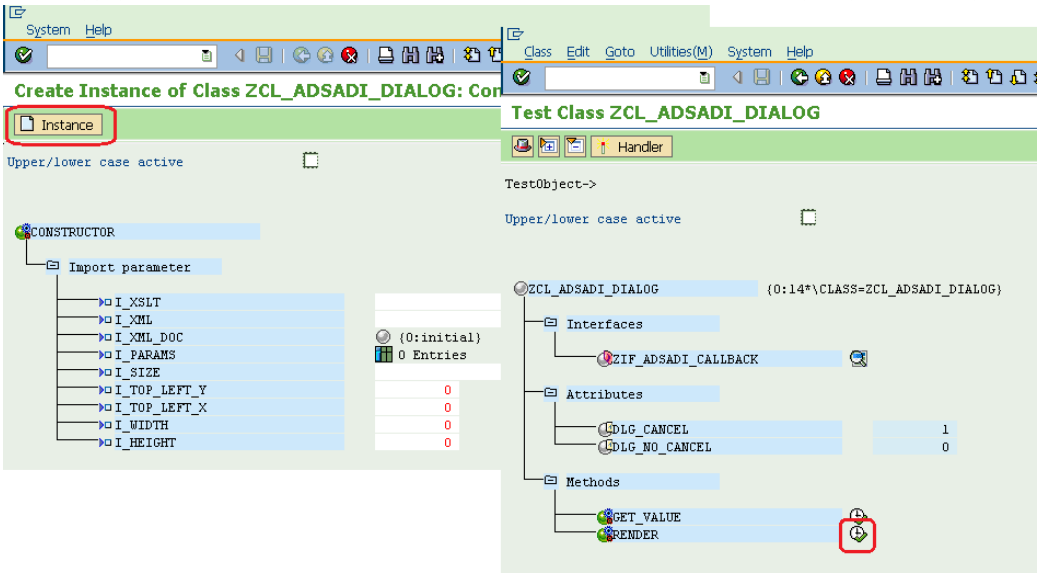
```
    bottom_right_x = i_top_left_x + i_width.
    bottom_right_y = i_top_left_y + i_height.
    call screen 100
      starting at i_top_left_x   i_top_left_y
      ending   at bottom_right_x bottom_right_y.
  endif.

  e_canceled = g_canceled.
```

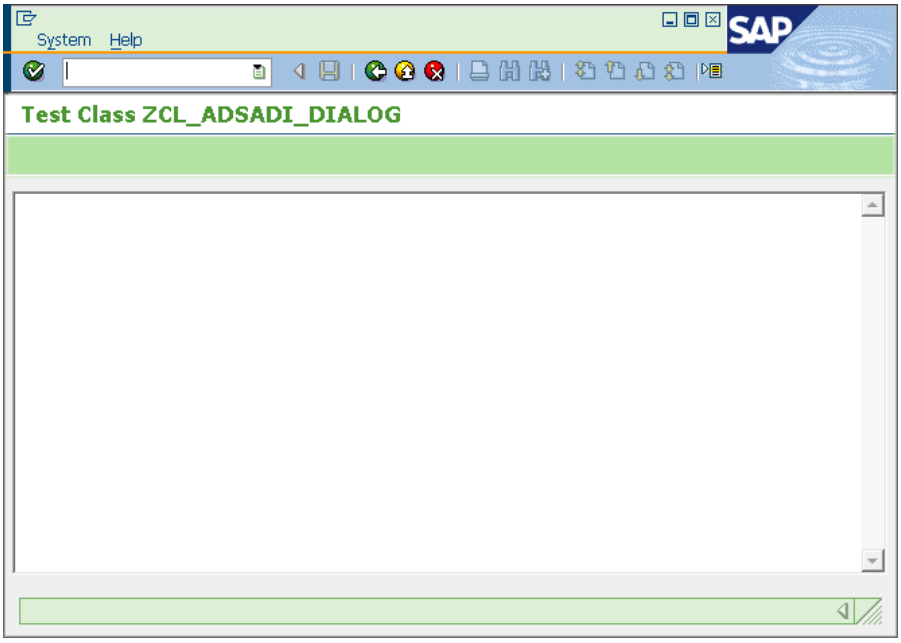
Source code top include of function module Z_ADSADI

```
function-pool zadsadi.                    "MESSAGE-ID ..

data: html_control type ref to cl_gui_html_viewer.
data: container type ref to cl_gui_custom_container.

data: g_callback type ref to zif_adsadi_callback.
data: g_canceled type i.
```

Test the dialog implementation







Back to screen 100, breathe life into it

Top include

```
data: myevent_tab TYPE cntl_simple_events.
data: myevent TYPE cntl_simple_event.
```

In the module include above status_0100, add an event handler for the HTML Viewer control:

```
class cl_myevent_handler definition.
  public section.
    methods: on_sapevent
      for event sapevent
      of cl_gui_html_viewer
```

```
      importing action frame getdata postdata query_table.
endclass.

class cl_myevent_handler implementation.

  method on_sapevent.
    if g_callback is bound.
      call method g_callback->action_post
        exporting
          action      = action
          getdata     = getdata
          postdata    = postdata
          query_table = query_table.
      if action is not initial.
        g_canceled = zif_adsadi_callback=>dlg_no_cancel.
      endif.
      perform close.
    endif.
  endmethod.                    "on_sapevent
endclass.

data: evt_receiver type ref to cl_myevent_handler.
```

Refactor out code to destroy the control

```
module user_command_0100 input.
  case sy-ucomm.
    when 'BACK'.
      perform close.
    when others.
      call method cl_gui_cfw=>dispatch.
  endcase.
endmodule.

form close.
  if not html_control is initial.
    call method html_control->free.
    free html_control.
    call method container->free.
    free container.
  endif.
  leave to screen 0.
endform.
```

Add code for event registration and page content loading

```
module status_0100 output.
  set pf-status 'ZADSADI'.
* SET TITLEBAR 'xxx'.
  if html_control is initial.
    create object container
      exporting
        container_name = 'CONTAINER'.

    create object html_control
      exporting
```

```
        parent = container.

*   register event
    myevent-eventid = html_control->m_id_sapevent.
    myevent-appl_event = 'x'.
    append myevent to myevent_tab.

    html_control->set_registered_events( myevent_tab ).
    create object evt_receiver.
    set handler evt_receiver->on_sapevent for html_control.

    perform load_page_content.
  endif.
endmodule.

form load_page_content.
  data: doc_url(80).
  data: html_tab type zadsadi_t_ml.
  data: html_l type line of zadsadi_t_ml.

  refresh html_tab.

  html_tab = g_callback->create_view( ).

  call method html_control->load_data
    exporting
      url          = doc_url
      size         = 0
      type         = 'text'
      subtype      = 'html'
    importing
      assigned_url = doc_url
    changing
      data_table   = html_tab
    exceptions
      others       = 1.

  if sy-subrc eq 0.
    call method html_control->show_url
      exporting
        url = doc_url.
  endif.
endform.
```

Implement ZCL_ADSADI_DIALOG methods derived from interface

Method action_post

```
  method zif_adsadi_callback~action_post.
    data: the_param type w3query.
    clear: me->query_table.
    " Gather post and get data
    if getdata is not initial.
      the_param-name = action.
```

```
    the_param-value = getdata.
    append the_param to me->query_table.
  endif.
  loop at query_table into the_param.
    append the_param to me->query_table.
  endloop.
endmethod.
```

Method create_view

```
method zif_adsadi_callback~create_view.
  data: ex type ref to  cx_xslt_runtime_error.
  data: ex2 type ref to cx_transformation_error.

  data: param type abap_trans_parmbind.

  data: begin of bgr,
        h1 type c length 2,
        h2 type c length 2,
        h3 type c length 2,
      end of bgr.

  data: hex type x length 3.
  data: buf type c length 10.
  data: color type i.

*   Background-Color1
  param-name = 'SAP_BACKGROUNDCOLOR1'.
  param-value = '#FFFFFF'.

  " Overwrite default with system value
  call method cl_gui_resources=>get_background_color
    exporting
      id     = cl_gui_resources=>col_background_level1
      state  = 0
    importing
      color  = color
    exceptions
      others = 1.
  if sy-subrc = 0.
    bgr = buf = hex = color.
    concatenate `#` bgr-h3 bgr-h2 bgr-h1 into param-value.
  endif.
  append param to me->xslt_params.

*   Background-Color2
  param-name = 'SAP_BACKGROUNDCOLOR2'.
  param-value = '#CCCCCC'.

  " Overwrite default with system value
  call method cl_gui_resources=>get_background_color
    exporting
```

```
      id     = cl_gui_resources=>col_background_level2
      state  = 0
    importing
      color  = color
    exceptions
      others = 1.
  if sy-subrc = 0.
    bgr = buf = hex = color.
    concatenate `#` bgr-h3 bgr-h2 bgr-h1 into param-value.
  endif.
  append param to me->xslt_params.


  try.
      if me->xml_doc is bound.
        call transformation (me->xslt)
                 parameters (me->xslt_params)
                 source xml me->xml_doc
                 result xml r.
      else.
        call transformation (me->xslt)
                 parameters (me->xslt_params)
                 source xml me->xml
                 result xml r.
      endif.
    catch: cx_xslt_runtime_error into ex.
    catch: cx_invalid_transformation into ex2.
  endtry.
endmethod.
```