File Handling utility using ALV Tree for Presentation and Application Server

Applies To:

SAP R/3 4.6

Summary

This Code sample involves the implementation of code for enabling browsing of presentation and application server files. It also enables File Transfer from the presentation server to application server through simple drag and drop functionality. The object is designed using the SAP ALV Tree. The Object implements two separate tree controls for both presentation and application servers. It also enables dynamic addition of nodes to existing displayed set of nodes. This functionality is achieved while the expander of a node is clicked.

By: Prabaharan Gnanasekaran

Company: Wipro Technologies

Date: 15 May 2006

Brief Description of Functionality:

The objects uses two Containers for implementing the two ALV Tree in the output. The standard code can be copied directly and used. Only adjustments to be done is create the screen with the screen number mentioned in program and in the created screen two containers needs to be created as mentioned in the program (CUSTCONT , CUSTCONT2).

The following options are available in the ALV Tree .

1) Double clicking functionality over a node in presentation server tree will open the file in the IE.

2) Context Menu functionality implemented separately for Folder type nodes and File type Nodes.

3) Drag and Drop of File From Presentation server to Application Server. Message to confirm the drag and drop function.

4) Dynamic Addition of nodes to existing Folder type nodes in both tree . An expander is placed by the side of every node of type Folder and when it is clicked the files inside that folder are added dynamically to that folder node. The expander is implemented by using the is_node_layout parameter present “Add_node” method .

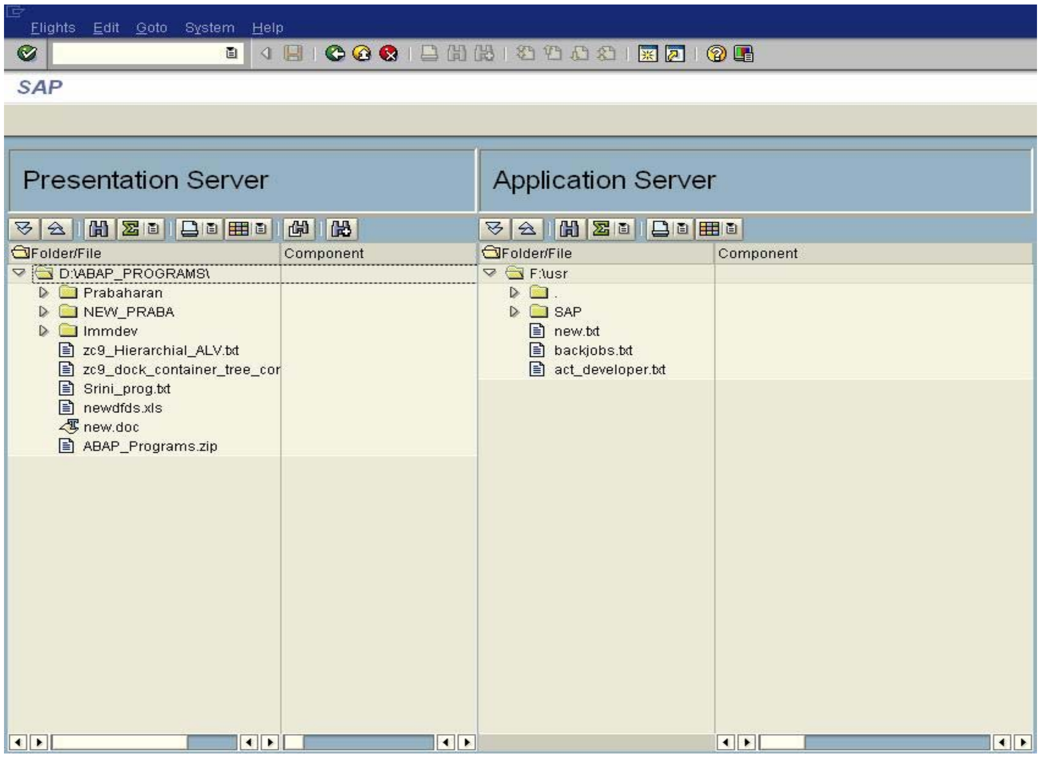
5) The messages used in the program are indicated with their corresponding number at the top of the code for easy reference.

6) The function module 'TMP_GUI_DIRECTORY_LIST_FILES' is used for fetching Presentation server files and the Unix Command ls – option is used to fetch files in Application server. The code is filled with ample inline comments to aid easy reference.

7) The structure ZC9_LINE used in the code is flat structure with one field named LINE of type character and length 350.

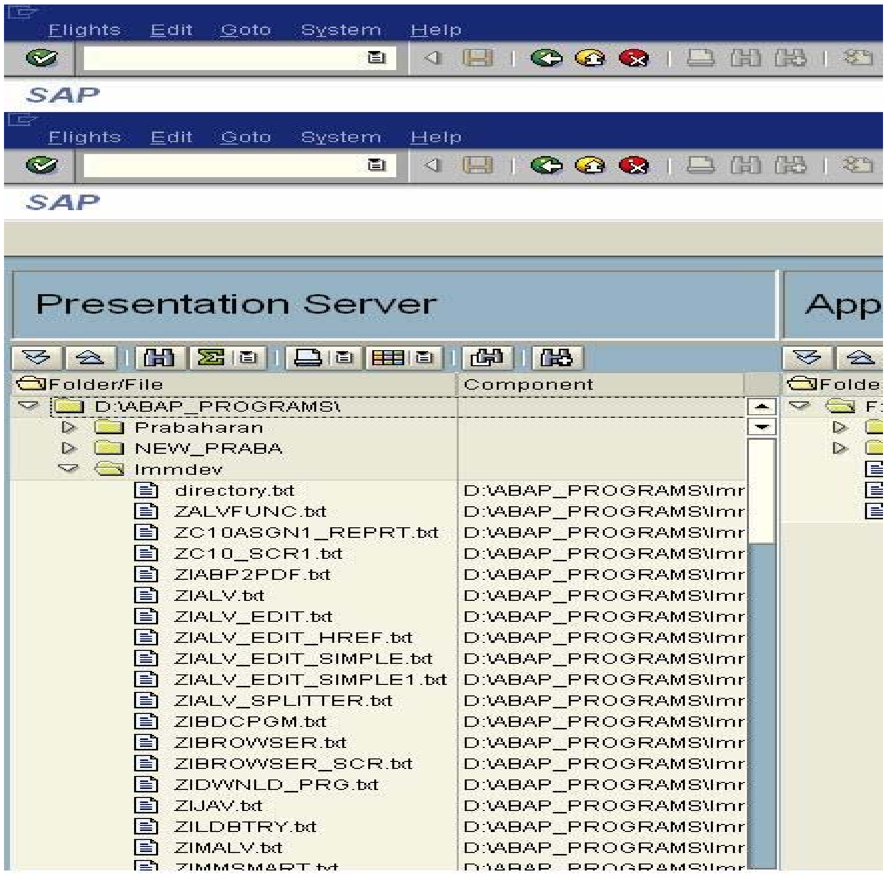
The Screen Shots of the Code Output :

1) General Output of the Program.
2) Dynamic addition of nodes to existing Parent Folder type nodes
3) Double clicking functionality to open files.
4) Drag and Drop functionality.
5) Context menu for file nodes for opening, transferring file contents to local machine or application server.



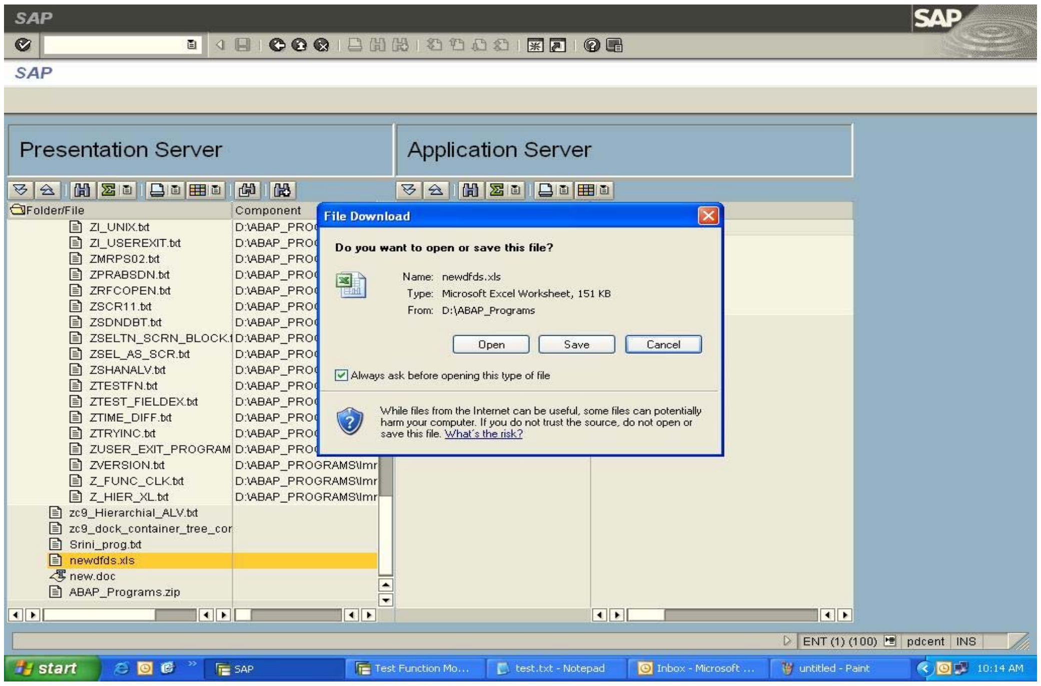
D:\ABAP_PROGRAMS\
Prabaharan
NEW_PRABA
Immdev
zc9_Hierarchial_ALV.txt
zc9_dock_container_tree_cor
Srini_prog.txt
newdfds.xls
new.doc
ABAP_Programs.zip

F:\usr
.
SAP
new.txt
backjobs.txt
act_developer.txt



Folder/File	Component
D:\ABAP_PROGRAMS\	
Prabaharan	
NEW_PRABA	
Immdev	
directory.txt	D:\ABAP_PROGRAMS\Imr
ZALVFUNC.txt	D:\ABAP_PROGRAMS\Imr
ZC10ASGN1_REPRT.txt	D:\ABAP_PROGRAMS\Imr
ZC10_SCR1.txt	D:\ABAP_PROGRAMS\Imr
ZIABP2PDF.txt	D:\ABAP_PROGRAMS\Imr
ZIALV.txt	D:\ABAP_PROGRAMS\Imr
ZIALV_EDIT.txt	D:\ABAP_PROGRAMS\Imr
ZIALV_EDIT_HREF.txt	D:\ABAP_PROGRAMS\Imr
ZIALV_EDIT_SIMPLE.txt	D:\ABAP_PROGRAMS\Imr
ZIALV_EDIT_SIMPLE1.txt	D:\ABAP_PROGRAMS\Imr
ZIALV_SPLITTER.txt	D:\ABAP_PROGRAMS\Imr
ZIBDCPGM.txt	D:\ABAP_PROGRAMS\Imr
ZIBROWSER.txt	D:\ABAP_PROGRAMS\Imr
ZIBROWSER_SCR.txt	D:\ABAP_PROGRAMS\Imr
ZIDWNLD_PRG.txt	D:\ABAP_PROGRAMS\Imr
ZIJAV.txt	D:\ABAP_PROGRAMS\Imr
ZILDBTRY.txt	D:\ABAP_PROGRAMS\Imr
ZIMALV.txt	D:\ABAP_PROGRAMS\Imr
ZIMMSMART.txt	D:\ABAP_PROGRAMS\Imr

© 2006 SAP AG The SAP Developer Network: http://sdn.sap.com



Presentation Server

Application Server

Folder/File	Component
ZI_UNIX.txt	D:\ABAP_PRO
ZI_USEREXIT.txt	D:\ABAP_PRO
ZMRPS02.txt	D:\ABAP_PRO
ZPRABSDN.txt	D:\ABAP_PRO
ZRFCOPEN.txt	D:\ABAP_PRO
ZSCR11.txt	D:\ABAP_PRO
ZSDNDBT.txt	D:\ABAP_PRO
ZSELTN_SCRN_BLOCK1	D:\ABAP_PRO
ZSEL_AS_SCR.txt	D:\ABAP_PRO
ZSHANALV.txt	D:\ABAP_PRO
ZTESTFN.txt	D:\ABAP_PRO
ZTEST_FIELDEX.txt	D:\ABAP_PRO
ZTIME_DIFF.txt	D:\ABAP_PRO
ZTRYINC.txt	D:\ABAP_PRO
ZUSER_EXIT_PROGRAM	D:\ABAP_PRO
ZVERSION.txt	D:\ABAP_PROGRAMS\Imr
Z_FUNC_CLK.txt	D:\ABAP_PROGRAMS\Imr
Z_HIER_XL.txt	D:\ABAP_PROGRAMS\Imr
zc9_Hierarchial_ALV.txt	
zc9_dock_container_tree_cor	
Srini_prog.txt	
newdfds.xls	
new.doc	
ABAP_Programs.zip	

File Download

Do you want to open or save this file?

Name: newdfds.xls
Type: Microsoft Excel Worksheet, 151 KB
From: D:\ABAP_Programs

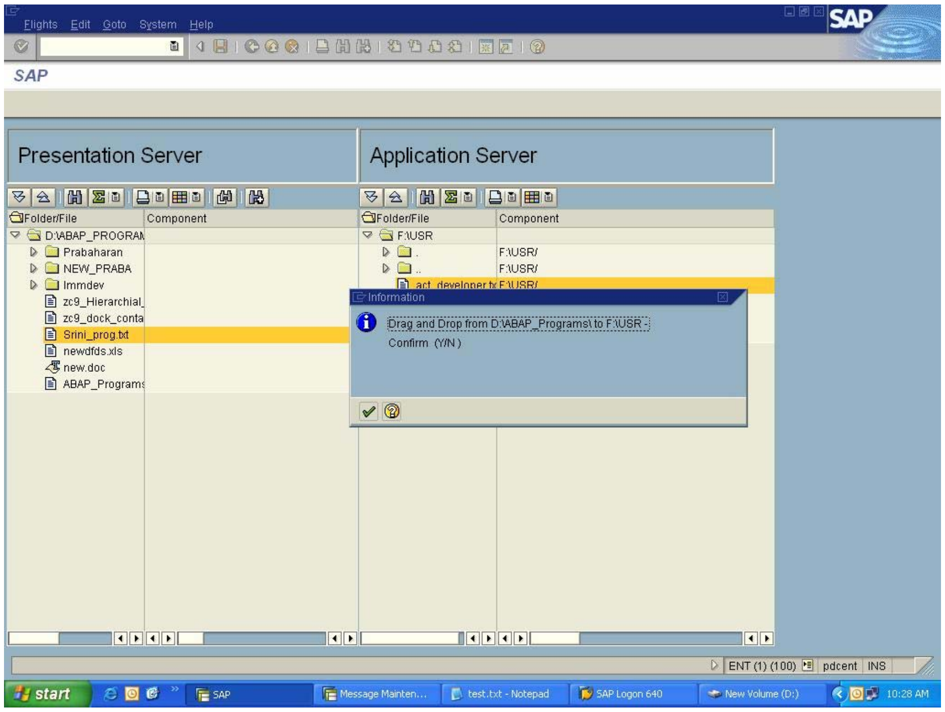
Open | Save | Cancel

☑ Always ask before opening this type of file

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

ENT (1) (100) | pdcent | INS

© 2006 SAP AG



Presentation Server

Folder/File	Component
D:\ABAP_PROGRAM	
Prabaharan	
NEW_PRABA	
Immdev	
zc9_Hierarchial_	
zc9_dock_conta	
Srini_prog.txt	
newdfds.xls	
new.doc	
ABAP_Programs	

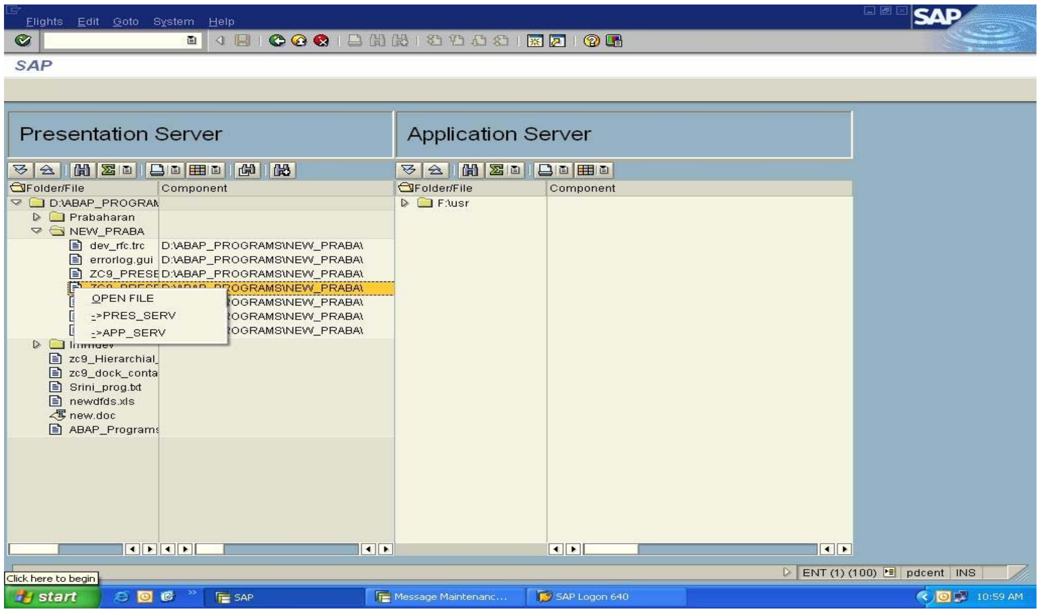
Application Server

Folder/File	Component
F:\USR	
.	F:\USR/
..	F:\USR/
act_developer.tx	F:\USR/

Information

Drag and Drop from D:\ABAP_Programs\ to F:\USR -

Confirm (Y/N)



Sample Code of the Program:

report zc9_presentation_serv_browse message-id zmc .

*&---

* Messages used in program for reference

*Message: 000 Cannot open the selected file

*   001   Records not found

*   003   Records deleted

*   037   Node not found in displayed tree

*   041   File Copy Error Access Denied

*   042   File Copied Successfully

```
*       005       Drag and Drop Function Done

*       013        File Copy Aborted by User

*&-----------------------------------------------------



Include <icon>.

Type-pools: slis.

type-pools: cntl.
tables: sscrfields.

data: outtabl type standard table of zc9_line1 with header line.
data: v_nodetext(100) type c.
data: v_act_grid type c.
data: fil_aptab type table of  sdokpath with header line,
      dir_aptab type table of  sdokpath with header line,
      fil_aptab1 type table of sdokpath with header line,
      fil_aptabstr type table of sdokpath ,
      dir_aptab1 type table of  sdokpath with header line.


data: begin of dat_tab occurs 3000,
      txt(180)              type c,
      end of dat_tab.

data: wa_tab like line of dat_tab.
data  v_apfile(110) type c.
data: dsn1(300) type c.

data: l_hierarchy_header type treev_hhdr.
data: l_hierarchy_header1 type treev_hhdr.

data: lt_events type cntl_simple_events,
       l_event type cntl_simple_event.
```

```
data: lt_events1 type cntl_simple_events.

data: fil_tab type table of  sdokpath with header line,
      dir_tab type table of  sdokpath with header line,
      fil_tab1 type table of sdokpath with header line,
      fil_tabstr type table of sdokpath ,
      dir_tab1 type table of  sdokpath with header line,
      g_toolbar type ref to cl_gui_toolbar.

*Define reference variables
data: g_alv_tree type ref to cl_gui_alv_tree,
      g_custom_container type ref to cl_gui_custom_container,
      gs_layout_tree type lvc_s_layo,
      g_dropeffect type i,
      g_handle_tree type i,
      g_handle_alv  type i,
      dragdrop_tree1 type ref to cl_dragdrop,
      dragdrop_tree2 type ref to cl_dragdrop,
       effect type i,
      gs_layout_alv  type lvc_s_layo,
      node_key type tv_nodekey.

*Define reference variables
data: g_alv_tree1 type ref to cl_gui_alv_tree,
      g_custom_container1 type ref to cl_gui_custom_container.

data:  it_tab type filetable,
      gd_subrc type i,
      answer type c.
data: selfolder(200) type c.
data: source_inp type string,
      dest_inp type string.

data: i_node_tabl type lvc_t_nkey.
data: ls_line type slis_listheader,
      pt_list_commentary type slis_t_listheader,
      l_logo type sdydo_value,
      v_input(120) type c,
      v_op_file(200) type c,
      v_strlen(3) type c,
      v_index like sy-tabix..

data:  no_file(3) type n,
      no_apfile(3) type n.

data: v_node_num type lvc_nkey,
      v_prev_node type lvc_nkey,
      v_node_apnum type lvc_nkey,
*       v_apincr_node type c,
      v_node_num1 type lvc_nkey,
      p_relat_key type lvc_nkey,
      node_key1 type lvc_nkey,
      v_node_key type lvc_t_nkey.

data: begin of node_det occurs 0,
      name(120) type c,
      node_num(12) type c,
      par_name(150) type c,
      end of node_det.

data: begin of node_apdet occurs 0,
      name(120) type c,
      node_num(12) type c,
      par_name(150) type c,
      end of node_apdet.

data: fil_tab2 like table of  node_det with header line.
data: fil_aptab2 like table of  node_det with header line.
data: wa_node_det like line of node_det.
data: wa_filtab like line of fil_tab2.
data: wa_apfiltab like line of fil_tab2.
data: l_node_text type lvc_value.
data  v_init type c.
data  v_apinit type c.
data  v_apincr_node type c value  ' '.
data  v_incr_node type c value  ' '.
```

© 2006 SAP AG

```
data  p_path1(150) type c.
data  p_apath1(150) type c.
data  tot_rec(3) type n.
data  tot_aprec(3) type n.
data : v_initx type c value 'X',
v_program(140) type c.
data: v_op_file1 like rlgrap-filename.
data: cmd like sscrfields-ucomm.


parameter: p_path(150) type c.
parameters: appserv like rlgrap-filename.

at selection-screen.
cmd = sscrfields-ucomm.


at selection-screen on value-request for p_path.

  call function 'TMP_GUI_BROWSE_FOR_FOLDER'
   exporting
     window_title          = 'Folder Select'
     initial_folder        = 'D:\'
   importing
     selected_folder       = p_path
* EXCEPTIONS
*   CNTL_ERROR             = 1
*   OTHERS                 = 2
            .
  if sy-subrc <> 0.
  endif.


*-----------------------------------------------------
*class for tool bar click.
*-----------------------------------------------------
class lcl_toolbar_event_recv definition.
```

© 2006 SAP AG

```
  public section.
    methods: on_function_selected
               for event function_selected of cl_gui_toolbar
                 importing fcode.
endclass.

*----------------------------------------------------
*class for tool bar click.
*----------------------------------------------------
class lcl_toolbar_event_recv implementation.

  method on_function_selected.
    data: lt_selected_nodes type lvc_t_nkey,
          l_selected_node   type lvc_nkey,
          l_rc              type c.

    case fcode.

      when 'SEARCH'.
        perform pop_n_search.

      when 'SEARCH+'.

        perform search_again.

    endcase.


  endmethod.
endclass.

*---------------------------------------------------------------------*
*       CLASS cl_tree_event_receiver1 DEFINITION
*---------------------------------------------------------------------*
class cl_tree_event_receiver1 definition.
  public section.

    methods handle_expand_apnc
    for event expand_nc of cl_gui_alv_tree
    importing node_key.

  methods handle_on_drop for event on_drop of cl_gui_alv_tree importing
        node_key
        drag_drop_object.


endclass.


*Class for Tree
*---------------------------------------------------------------------*
*       CLASS cl_tree_event_receiver DEFINITION
*---------------------------------------------------------------------*
class cl_tree_event_receiver definition.
  public section.

    methods handle_expand_nc
    for event expand_nc of cl_gui_alv_tree
    importing node_key.


    methods handle_node_context_menu_rq for event
    node_context_menu_request of cl_gui_alv_tree importing node_key menu
 .

    methods handle_node_cm_sel
    for event node_context_menu_selected of cl_gui_alv_tree
    importing  fcode sender.

    methods handle_node_keypress for event node_keypress of
    cl_gui_alv_tree importing node_key key.

    methods handle_node_dbclick for event node_double_click of
    cl_gui_alv_tree importing node_key .

*   Drag
```

```
    methods handle_on_drag
      for event on_drag of cl_gui_alv_tree
      importing node_key
                fieldname
                drag_drop_object.

*Drop Complete
   methods handle_dd_comp for event on_drop_complete of cl_gui_alv_tree
    importing node_key
              fieldname
              drag_drop_object.
endclass.


*---------------------------------------------------------------------*
*       CLASS cl_tree_event_receiver IMPLEMENTATION
*---------------------------------------------------------------------*
class cl_tree_event_receiver1 implementation.

  method handle_expand_apnc.
    v_node_apnum = node_key .
    v_node_apnum = v_node_apnum - 1.
    if v_apincr_node = 'X'.
      v_apincr_node = ' '.
    endif.
    perform fetch_apfile_name.
* §5. Send data to frontend.
    call method g_alv_tree1->frontend_update.
  endmethod.


  method handle_on_drop.
*v_node_apnum = node_key .
*    v_node_apnum = v_node_apnum + 1.
*    read table node_apdet with key node_num =  v_node_apnum .
*    concatenate node_apdet-par_name '/'  node_apdet-name  into appserv.
*    v_node_apnum = v_node_apnum - 1.
```

```
  endmethod.

endclass.


*---------------------------------------------------------------------*
*       CLASS cl_tree_event_receiver IMPLEMENTATION
*---------------------------------------------------------------------*
class cl_tree_event_receiver implementation.
  method handle_node_keypress.
    message i000(zmc).
*    perform display_users .
  endmethod.

  method handle_node_dbclick .
    v_node_num = node_key.
    perform double_click.
  endmethod.

  method handle_expand_nc.
    v_node_num = node_key .
    v_node_num = v_node_num - 1.
    if v_incr_node = 'X'.
      v_incr_node = ' '.
    endif.
    perform fetch_file_name.
* §5. Send data to frontend.
    call method g_alv_tree->frontend_update.
  endmethod.


  method handle_node_context_menu_rq.
    move node_key to node_key1.
* In this case the standard menu is cleared.
    call method menu->clear.
* The next line defines one line of the context menu.
    read table fil_tab2 into wa_filtab with key node_num = node_key1.
    if wa_filtab-name ns '.'.
```

```
      call method menu->add_function
        exporting
          fcode = 'EXP_NODE'
          text  = 'Expand Node'.
    else.

      call method menu->add_function
        exporting
          fcode = 'OPEN'
          text  = 'OPEN FILE'.



      call method menu->add_function
        exporting
          fcode = 'TRANS'
          text  = '->PRES_SERV'.

      if not appserv is  initial.
        call method menu->add_function
          exporting
            fcode = 'TRANS1'
            text  = '->APP_SERV'.
      endif.
    endif.
    clear wa_filtab.
  endmethod.

*--------------------------------------------------------------
  method handle_node_cm_sel.
* At this point of execution, the user selected a menu entry of the
* menu build up in event handler method handle_node_cm_req.
* Query your own function codes and react accordingly.
    data l_rc type c.

    case fcode.
      when 'EXP_NODE'.
        call method g_alv_tree->expand_node
```

```
          exporting
            i_node_key          = node_key1
*     I_LEVEL_COUNT       = 1
*     I_EXPAND_SUBTREE    =
*   EXCEPTIONS
*     FAILED              = 1
*     ILLEGAL_LEVEL_COUNT = 2
*     CNTL_SYSTEM_ERROR   = 3
*     NODE_NOT_FOUND      = 4
*     CANNOT_EXPAND_LEAF  = 5
*     others              = 6
                .
        if sy-subrc <> 0.

        endif.


      when 'TRANS'.
        perform trans_file.

      when 'OPEN'.
        v_node_num  = node_key1.
        perform double_click.

      when 'TRANS1'.
        v_node_num  = node_key1.

        perform trans_app.

        write:/ 'done'.
*-----------------------------------------------------
*v_node_num = '&VIRTUALROOT'.
* call method g_alv_tree->get_subtree
*   exporting
*     i_node_key       = v_node_num
*   importing
*     et_subtree_nodes = i_node_tabl.
*     .
*       WRITE:/ 'READ NODE'.
*---------------------------------------------------
    endcase.
  endmethod.

* Drag & Drop
  method handle_on_drag.
    message e005(zmc).
  endmethod.

  method handle_dd_comp.
  endmethod.

endclass.



start-of-selection.

  p_path1 = p_path.
  p_apath1 = appserv.
  call screen 200.

*&---------------------------------------------------------------------*
*&      Form  tree_intial
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
form tree_intial.

  data: l_tree_container_name(30) type c.

  l_tree_container_name = 'CUSTCONT'.

  create object g_custom_container
      exporting
            container_name = l_tree_container_name
```

© 2006 SAP AG

```
      exceptions
         cntl_error                  = 1
         cntl_system_error           = 2
         create_error                = 3
         lifetime_error              = 4
         lifetime_dynpro_dynpro_link = 5.
  if sy-subrc <> 0.
    message x208(00) with 'ERROR'(100).
  endif.

* create tree control
  create object g_alv_tree
    exporting
        parent              = g_custom_container
        node_selection_mode = cl_gui_column_tree=>node_sel_mode_single
        item_selection      = ''
        no_html_header      = ''
        no_toolbar          = ''
    exceptions
        cntl_error                   = 1
        cntl_system_error            = 2
        create_error                 = 3
        lifetime_error               = 4
        illegal_node_selection_mode  = 5
        failed                       = 6
        illegal_column_name          = 7.
  if sy-subrc <> 0.
    message x208(00) with 'ERROR'.                          "#EC NOTEXT
  endif.

*For hierarchy building

  l_hierarchy_header-t_image = '000001'.
  l_hierarchy_header-heading = 'Folder/File'.
  l_hierarchy_header-tooltip = 'Folder/File'.
  l_hierarchy_header-width = 30.
  l_hierarchy_header-width_pix = ' '.
```

© 2006 SAP AG

```
* LIST HEADING LINE: TYPE H
  clear ls_line.
  ls_line-typ  = 'H'.
  ls_line-info = 'Presentation Server'.
  append ls_line to pt_list_commentary.


  gs_layout_tree-s_dragdrop-row_ddid = g_handle_tree.


*Create empty Tree Control with no values in table
  call method g_alv_tree->set_table_for_first_display
     exporting
                i_structure_name     = 'SDOKPATH'
                it_list_commentary          = pt_list_commentary
*                i_logo                      = l_logo
                is_hierarchy_header  = l_hierarchy_header
             changing
             it_outtab           = fil_tabstr."table must be empty

*Event receiver decl for Tree.
  data: tree_event_receiver type ref to cl_tree_event_receiver.

*Event recvr for tool bar.
*The method inside this class is for class cl_gui_object . This class
*is subclass of cl_gui_object which is also a super class for
*cl_gui_alv_tree.This is how these two classes are linked
  data: l_event_receiver type ref to lcl_toolbar_event_recv.


*Before using the events we need to register those events first.
  data: lt_events type cntl_simple_events,
         l_event type cntl_simple_event.
  clear l_event.

  l_event-eventid = cl_gui_column_tree=>eventid_expand_no_children.
*cl_tree_control_base
```

```
  append l_event to lt_events.

*the events below is in cl_tree_control_base class but
*cl_gui_column_tree is a subclass of it and hence inherits all its
*events
  append l_event to lt_events.
  l_event-eventid = cl_gui_column_tree=>eventid_node_context_menu_req.
  append l_event to lt_events.

  l_event-eventid = cl_gui_column_tree=>eventid_node_keypress.
  append l_event to lt_events.
  l_event-eventid = cl_gui_column_tree=>eventid_node_double_click.
  append l_event to lt_events.

*Event recver object for tree.
*The event recvr is created as an object for the class which implements
*the methods for ALV tree.
  create object tree_event_receiver.

* register events
  call method g_alv_tree->set_registered_events
    exporting
      events                    = lt_events
    exceptions
      cntl_error                = 1
      cntl_system_error         = 2
      illegal_event_combination = 3.
  if sy-subrc <> 0.

    message e003(zmc).

  endif.

* set handler for tree1 events
*Set handler event recvr object---> method name for alv tree object name
*this is how event recvr and alv tree methods are linked.
  set handler tree_event_receiver->handle_expand_nc for g_alv_tree.
  set handler tree_event_receiver->handle_node_context_menu_rq
             for g_alv_tree.
  set handler tree_event_receiver->handle_node_cm_sel
              for g_alv_tree.
  set handler tree_event_receiver->handle_node_keypress
              for g_alv_tree.
  set handler tree_event_receiver->handle_node_dbclick
              for g_alv_tree.
  set handler tree_event_receiver->handle_on_drag for g_alv_tree.

  set handler tree_event_receiver->handle_dd_comp for g_alv_tree.



*This add button should be called before set handler or else g_toolbar
*will be initial and lead to shortdump
  perform add_button.

*Handler for tool bar.
  create object l_event_receiver.
  set handler l_event_receiver->on_function_selected for g_toolbar.

  perform init_dragdrop.

  perform fetch_list.
* §5. Send data to frontend.
  call method g_alv_tree->frontend_update.

  v_init = '2'.

endform.                    " tree_intial


*&---------------------------------------------------------------------*
*&      Form  add_this_as_leaf
*&---------------------------------------------------------------------*
form add_this_as_leaf using    p_fil_tab-pathname  changing
p_v_node_num p_v_node_num1 .
```

```
data: l_node_layout type lvc_s_layn.

data: l_node_text type lvc_value.
l_node_text = p_fil_tab-pathname.

if sy-tabix gt no_file.
  l_node_layout-expander  = 'X'.
endif.
l_node_layout-dragdropid = g_handle_tree.

if p_fil_tab-pathname cs '.MP3'.
  l_node_layout-exp_image = icon_voice_output.
  l_node_layout-n_image = icon_voice_output.
elseif p_fil_tab-pathname cs '.DOC'.
  l_node_layout-exp_image = icon_word_processing.
  l_node_layout-n_image = icon_word_processing.
elseif p_fil_tab-pathname cs '.PDF'.
  l_node_layout-exp_image = icon_pdf.
  l_node_layout-n_image = icon_pdf.
elseif p_fil_tab-pathname cs '.JPG'.
  l_node_layout-exp_image = icon_jpg.
  l_node_layout-n_image = icon_jpg.
elseif p_fil_tab-pathname cs '.HTML'.
  l_node_layout-exp_image = icon_htm.
  l_node_layout-n_image = icon_htm.
elseif l_node_layout-expander  <> 'X'.
endif.

call method g_alv_tree->add_node
    exporting
          i_relat_node_key = p_v_node_num
          i_relationship   = cl_gui_column_tree=>relat_first_child
          i_node_text      = l_node_text
         is_node_layout = l_node_layout
          importing
          e_new_node_key = p_v_node_num1.
  if sy-tabix gt no_file.
    wa_node_det-name = p_fil_tab-pathname.
    wa_node_det-node_num = p_v_node_num1.
    wa_node_det-par_name = p_path.
    append wa_node_det to node_det.
  endif.

  wa_node_det-name = p_fil_tab-pathname.
  wa_node_det-node_num = p_v_node_num1.
  wa_node_det-par_name = p_path.
  append wa_node_det to fil_tab2.

  clear l_node_layout.
  if  v_init eq '1'.
    p_v_node_num = 1.
  else.
    p_v_node_num = v_node_num.
  endif.
  clear l_node_layout.
endform.                    " add_this_as_leaf


*&---------------------------------------------------------------------*
*&      Form  add_this_as_leaf
*&---------------------------------------------------------------------*
form add_this_as_leaf1 using    p_fil_tab-pathname  p_relat_key
                      changing p_v_node_num .

  data: l_node_layout type lvc_s_layn.

  data: l_node_text type lvc_value.
  l_node_text = p_fil_tab-pathname.

  if sy-tabix gt no_file.
    l_node_layout-expander  = 'X'.
  endif.

  call method g_alv_tree->add_node
```

© 2006 SAP AG

```
    exporting
          i_relat_node_key = p_relat_key
          i_relationship   = cl_gui_column_tree=>relat_first_child
          i_node_text      = l_node_text
         is_outtab_line = fil_tab-pathname
         is_node_layout = l_node_layout
          importing
          e_new_node_key = p_v_node_num.

  clear l_node_layout.
endform.                    " add_this_as_leaf

*&---------------------------------------------------------------------*
*&      Module  STATUS_0200  OUTPUT
*&---------------------------------------------------------------------*
module status_0200 output.
  set pf-status 'NEWSTAT'.

  if g_alv_tree is initial.
    perform tree_intial.
  endif.

  if g_alv_tree1 is initial.
    perform tree_initial1.
  endif.

endmodule.                 " STATUS_0200  OUTPUT
*&---------------------------------------------------------------------*
*&      Form  fetch_file_name
*&---------------------------------------------------------------------*
form fetch_file_name.

  describe table fil_tab lines tot_rec .

  if v_node_num le tot_rec.
    read table fil_tab index v_node_num.
    concatenate p_path1 '\' fil_tab-pathname '\' into p_path.
  else.
```

```
    v_node_num = v_node_num + 1.
    read table node_det with key node_num =  v_node_num .
    concatenate node_det-par_name '\'  node_det-name '\' into p_path.
    v_node_num = v_node_num - 1.
  endif.
  replace '\\' with '\' into p_path.
  perform fetch_list.

endform.                    " fetch_file_name
*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0200  INPUT
*&---------------------------------------------------------------------*
module user_command_0200 input.
  data: ok_code type sy-ucomm.
  case ok_code.
    when '&F03'.
      set screen 0.
      leave screen.

    when 'TRAN'.
      perform trans_file.
  endcase.
endmodule.
" USER_COMMAND_0200  INPUT
*&---------------------------------------------------------------------*
*&      Form  fetch_list
*&---------------------------------------------------------------------*
form fetch_list.

*Now start creating the hierarchy and filling the table.

  if ( fil_tab[] is initial ) and  ( dir_tab[] is initial ).
    concatenate p_path '\' into p_path.
    call function 'TMP_GUI_DIRECTORY_LIST_FILES'
      exporting
        directory       = p_path
*   FILTER           = '*.*'
* IMPORTING
```

© 2006 SAP AG

```
*   FILE_COUNT       =
*   DIR_COUNT        =
      tables
        file_table       = fil_tab
        dir_table        = dir_tab
   exceptions
     cntl_error       = 1
     others           = 2
            .
    if sy-subrc <> 0.
    endif.


    if ( fil_tab[] is initial ) and  ( dir_tab[] is initial ).
      message e001(zmc).
      exit.
    endif.

  else.
    call function 'TMP_GUI_DIRECTORY_LIST_FILES'
        exporting
          directory        = p_path
*   FILTER           = '*.*'
* IMPORTING
*   FILE_COUNT       =
*   DIR_COUNT        =
        tables
          file_table       = fil_tab1
          dir_table        = dir_tab1
     exceptions
       cntl_error       = 1
       others           = 2
              .
    if sy-subrc <> 0.
    endif.

  endif.
```

```
if ( fil_tab1[] is initial ) and ( dir_tab1[] is initial ) and v_init <>
                 '2' .
    fil_tab1[] = fil_tab[].
    dir_tab1[] = dir_tab[].
    append lines of dir_tab to fil_tab.
    clear v_init.
  endif.
  describe table fil_tab1 lines no_file.
  append lines of dir_tab1 to fil_tab1.


  if v_init is initial.
    perform add_this_as_leaf1 using p_path p_relat_key changing
    v_node_num .
    v_init = '1'.
  endif.

  loop at fil_tab1.
    if v_init ne '2'.
   perform add_this_as_leaf using fil_tab1-pathname changing v_node_num
                         v_node_num1 .
    else.
      if v_incr_node is initial.
        v_node_num = v_node_num + 1.
        v_incr_node = 'X'.
      endif.
      perform add_this_as_leafn using fil_tab1-pathname  v_node_num
      changing v_node_num1 .
    endif.
  endloop.

endform.                    " fetch_list


*&---------------------------------------------------------------------*
*&      Form  add_this_as_leaf
*&---------------------------------------------------------------------*
```

```
form add_this_as_leafn using    p_fil_tab-pathname
p_v_node_num  changing p_v_node_num1 .

  data: l_node_layout type lvc_s_layn.

  data: l_node_text type lvc_value.
  l_node_text = p_fil_tab-pathname.

  if sy-tabix gt no_file.
    l_node_layout-expander  = 'X'.
  endif.

  l_node_layout-dragdropid = g_handle_tree.

  if p_fil_tab-pathname cs '.MP3' or p_fil_tab-pathname cs '.RM'.
    l_node_layout-exp_image = icon_voice_output.
    l_node_layout-n_image = icon_voice_output.
elseif p_fil_tab-pathname cs '.WMV' or p_fil_tab-pathname cs '.AVI'  or
                p_fil_tab-pathname cs '.MPG'.
    l_node_layout-exp_image = icon_video.
    l_node_layout-n_image = icon_video.
  elseif p_fil_tab-pathname cs '.DOC'.
    l_node_layout-exp_image = icon_word_processing.
    l_node_layout-n_image = icon_word_processing.
  elseif p_fil_tab-pathname cs '.PDF'.
    l_node_layout-exp_image = icon_pdf.
    l_node_layout-n_image = icon_pdf.
  elseif p_fil_tab-pathname cs '.JPG'.
    l_node_layout-exp_image = icon_jpg.
    l_node_layout-n_image = icon_jpg.
  elseif p_fil_tab-pathname cs '.HTML'.
    l_node_layout-exp_image = icon_htm.
    l_node_layout-n_image = icon_htm.
  elseif p_fil_tab-pathname cs '.PST'.
    l_node_layout-exp_image = icon_mail.
    l_node_layout-n_image = icon_mail.
  elseif l_node_layout-expander  <> 'X'.
```

© 2006 SAP AG

```
  endif.


  call method g_alv_tree->add_node
     exporting
          i_relat_node_key = p_v_node_num
        i_relationship   = cl_gui_column_tree=>relat_last_child
        is_outtab_line = p_path
          i_node_text      = l_node_text
         is_node_layout = l_node_layout
          importing
          e_new_node_key = p_v_node_num1.

  if sy-tabix gt no_file.
    wa_node_det-name = p_fil_tab-pathname.
    wa_node_det-node_num = p_v_node_num1.
    wa_node_det-par_name = p_path.
    append wa_node_det to node_det.
  endif.

  wa_node_det-name = p_fil_tab-pathname.
  wa_node_det-node_num = p_v_node_num1.
  wa_node_det-par_name = p_path.
  append wa_node_det to fil_tab2.

  clear l_node_layout.
endform.                    " add_this_as_leaf
*&---------------------------------------------------------------------*
*&      Form  add_button
*&---------------------------------------------------------------------*
form add_button.

  call method g_alv_tree->get_toolbar_object
          importing
              er_toolbar = g_toolbar.

  check not g_toolbar is initial. "could happen if you do not use the
  "standard toolbar
```

```
* §2.Modify toolbar with methods of CL_GUI_TOOLBAR:
* add seperator to toolbar
  call method g_toolbar->add_button
          exporting
              fcode     = ''
              icon      = ''
              butn_type = cntb_btype_sep.

* add Standard Button to toolbar (for SEARCH)
  call method g_toolbar->add_button
          exporting
              fcode     = 'SEARCH'
              icon      = '@JH@'
              butn_type = cntb_btype_button
              text      = ''
              quickinfo = text-901.

  call method g_toolbar->add_button
          exporting
              fcode     = ''
              icon      = ''
              butn_type = cntb_btype_sep.

* add Standard Button to toolbar (for Delete Subtree)
  call method g_toolbar->add_button
          exporting
              fcode     = 'SEARCH+'
             icon      = '@4E@'
              butn_type = cntb_btype_button
              text      = ''
              quickinfo = text-901.   "Delete subtree

endform.                    " add_button
*&---------------------------------------------------------------------*
*&      Form  search_file
*&---------------------------------------------------------------------*
form search_file.
```

```
  clear: v_node_key[], v_node_key.
  append wa_filtab-node_num to v_node_key.

  call method g_alv_tree->set_selected_nodes
    exporting
      it_selected_nodes       = v_node_key
    exceptions
      cntl_system_error       = 1
      dp_error                = 2
      failed                  = 3
      error_in_node_key_table = 4
      others                  = 5
          .
  if sy-subrc <> 0.
    message i037(zmc).
  endif.

endform.                    " search_file
*&---------------------------------------------------------------------*
*&      Form  pop_n_search
*&---------------------------------------------------------------------*
form pop_n_search.

  clear v_input.
  call function 'POPUP_TO_GET_VALUE'
    exporting
      fieldname                 = 'FILENAME'
      tabname                   = 'RLGRAP'
      titel                     = 'INPUT'
      valuein                   = v_input
   importing
*         ANSWER                    = v_input
     valueout                  = v_input
*       EXCEPTIONS
*         FIELDNAME_NOT_FOUND       = 1
*         OTHERS                    = 2
            .
  if sy-subrc <> 0.

  endif.

  if sy-subrc = 0.
    concatenate v_input '*' into v_input.
    search fil_tab2 for v_input and mark.
    read table fil_tab2 into wa_filtab index sy-tabix.

    if sy-subrc = 0.
      v_index = sy-tabix.
      clear: v_node_key[], v_node_key.

      append wa_filtab-node_num to v_node_key.
      perform search_file.
    else.
      message i037(zmc).
    endif.
  endif.


endform.                    " pop_n_search
*&---------------------------------------------------------------------*
*&      Form  search_again
*&---------------------------------------------------------------------*
form search_again.
  v_index = v_index + 1.
  search fil_tab2 for v_input starting at v_index  and mark.
  read table fil_tab2 into wa_filtab index sy-tabix.
  v_index = sy-tabix.
  if sy-subrc = 0.
    clear: v_node_key[], v_node_key.
    append wa_filtab-node_num to v_node_key.
    perform search_file.
  else.
    message i037(zmc).
  endif.
endform.                    " search_again
```

```
*&---------------------------------------------------------------------*
*&      Form  trans_file
*&---------------------------------------------------------------------*
form trans_file.

  data: i_sel_nodes type lvc_t_nkey,
        sel_node_no(12) type c.

  call method g_alv_tree->get_selected_nodes
    changing
      ct_selected_nodes = i_sel_nodes
*  EXCEPTIONS
*    CNTL_SYSTEM_ERROR = 1
*    DP_ERROR          = 2
*    FAILED            = 3
*    others            = 4
          .
  if sy-subrc <> 0.
  endif.

*CALL SELECTION-SCREEN 500 STARTING AT 10 10.
  read table i_sel_nodes index 1 into sel_node_no.
  v_node_num = sel_node_no.
  read table fil_tab2 into wa_filtab with key node_num = v_node_num.
  concatenate wa_filtab-par_name '\' wa_filtab-name into v_op_file.
  replace '\\' with '\' into v_op_file.
  clear i_sel_nodes[].



  call function 'TMP_GUI_BROWSE_FOR_FOLDER'
   exporting
     window_title          = 'Folder'
     initial_folder        = 'D:\'
   importing
     selected_folder       = selfolder
* EXCEPTIONS
*   CNTL_ERROR            = 1
*   OTHERS                 = 2
          .
  if sy-subrc <> 0.

  endif.

  if not selfolder is initial.
    source_inp = v_op_file.
    concatenate selfolder '\' wa_filtab-name into selfolder.
    dest_inp = selfolder.
    call method cl_gui_frontend_services=>file_copy
      exporting
        source             = source_inp
        destination        = dest_inp
**    OVERWRITE          = SPACE
      exceptions
        cntl_error         = 1
        error_no_gui       = 2
        wrong_parameter    = 3
        disk_full          = 4
        access_denied      = 5
        file_not_found     = 6
        destination_exists = 7
        unknown_error      = 8
        path_not_found     = 9
        disk_write_protect = 10
        drive_not_ready    = 11
        others             = 12
        .
    if sy-subrc <> 0.
      message e041(zmc).
    else.
      message i042(zmc).
    endif.

  endif.

endform.                    " trans_file
```

© 2006 SAP AG

```
*&---------------------------------------------------------------------*
*&      Form  double_click
*&---------------------------------------------------------------------*
form double_click.
  read table fil_tab2 into wa_filtab with key node_num = v_node_num.
  concatenate wa_filtab-par_name '\' wa_filtab-name into v_op_file.
  replace '\\' with '\' into v_op_file.
  v_strlen = strlen( v_op_file ).

  v_strlen = v_strlen - 3.

  case  v_op_file+v_strlen(3) .
    when 'mp3'.
      v_program =  'C:\Program Files\Winamp\winamp.exe'.
    when 'doc' or 'DOC'.
      v_program = 'C:\Program Files\Internet Explorer\IEXPLORE.EXE'.
    when 'tml' or 'TML'.
      v_program = 'C:\Program Files\Internet Explorer\IEXPLORE.EXE'.
    when 'jpg' or 'JPG'.
      v_program = 'C:\Program Files\Internet Explorer\IEXPLORE.EXE'.
    when 'wmv' or 'WMV' or 'mpg' or 'MPG'.
      v_program = 'C:\Program Files\Windows Media Player\wmplayer.exe'.
    when others.
      v_program = 'C:\Program Files\Internet Explorer\IEXPLORE.EXE'.
  endcase.

  call function 'WS_EXECUTE'
        exporting
             program = v_program
             commandline     = v_op_file
*            INFORM         = ''
           exceptions
             prog_not_found = 1.
  if sy-subrc <> 0.
    message i000(zmc).
  endif.
  clear: v_op_file , v_program, wa_filtab,v_node_num.
```

```
endform.                    " double_click
*&---------------------------------------------------------------------*
*&      Form  trans_app
*&---------------------------------------------------------------------*
form trans_app.

 if appserv1 is initial.
    appserv1 = appserv.
    endif.

    call selection-screen 500 starting at 10 10.

*    at selection-screen.
*       data: cmd like sscrfields-ucomm.
*        cmd = sscrfields-ucomm.
*    endat.

if cmd = 'CRET'.
  read table fil_tab2 into wa_filtab with key node_num = v_node_num.
  concatenate wa_filtab-par_name wa_filtab-name into v_op_file1.

  concatenate appserv '/' wa_filtab-name into dsn1.
  translate dsn1 to lower case.
  call function 'WS_UPLOAD'
   exporting
*   CODEPAGE                      = ' '
     filename                     = v_op_file1
     filetype                     = 'ASC'
*   HEADLEN                       = ' '
*   LINE_EXIT                     = ' '
*   TRUNCLEN                      = ' '
*   USER_FORM                     = ' '
*   USER_PROG                     = ' '
*   DAT_D_FORMAT                  = ' '
* IMPORTING
*   FILELENGTH                    =
    tables
```

```
      data_tab                    = dat_tab
* EXCEPTIONS
*   CONVERSION_ERROR              = 1
*   FILE_OPEN_ERROR               = 2
*   FILE_READ_ERROR               = 3
*   INVALID_TYPE                  = 4
*   NO_BATCH                      = 5
*   UNKNOWN_ERROR                 = 6
*   INVALID_TABLE_WIDTH           = 7
*   GUI_REFUSE_FILETRANSFER       = 8
*   CUSTOMER_ERROR                = 9
*   OTHERS                        = 10
            .
  if sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.

  else.
    open dataset dsn1 for output in text mode. " encoding default.
    if sy-subrc = 0.
      loop at dat_tab into wa_tab.
        transfer wa_tab to dsn1.
      endloop.
      close dataset dsn1.
      if sy-subrc = 0.
      message i009(zmc).
      endif.
    endif.
  endif.
else.
message i013(ZMC).
endif.
endform.                    " trans_app
*&---------------------------------------------------------------------*
*&      Form  tree_initial1
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
```

```
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form tree_initial1.

  data: l_tree_container_name(30) type c.

  l_tree_container_name = 'CUSTCONT2'.

  create object g_custom_container1
      exporting
            container_name = l_tree_container_name
      exceptions
            cntl_error                  = 1
            cntl_system_error           = 2
            create_error                = 3
            lifetime_error              = 4
            lifetime_dynpro_dynpro_link = 5.
  if sy-subrc <> 0.
    message x208(00) with 'ERROR'(100).
  endif.

* create tree control
  create object g_alv_tree1
    exporting
        parent              = g_custom_container1
        node_selection_mode = cl_gui_column_tree=>node_sel_mode_single
        item_selection      = ''
        no_html_header      = ' '
        no_toolbar          = ''
    exceptions
        cntl_error                  = 1
        cntl_system_error           = 2
        create_error                = 3
        lifetime_error              = 4
        illegal_node_selection_mode  = 5
        failed                      = 6
        illegal_column_name         = 7.
  if sy-subrc <> 0.
    message x208(00) with 'ERROR'.                         "#EC NOTEXT
  endif.

  clear: pt_list_commentary[],pt_list_commentary.

  clear ls_line.
  ls_line-typ  = 'H'.
  ls_line-info = 'Application Server'.
  append ls_line to pt_list_commentary.

  clear fil_tabstr[].
  clear fil_tabstr.
  l_hierarchy_header1 = l_hierarchy_header .
*Create empty Tree Control with no values in table
  call method g_alv_tree1->set_table_for_first_display
     exporting
              i_structure_name     = 'SDOKPATH'
              it_list_commentary          = pt_list_commentary
*              i_logo                     = l_logo
              is_hierarchy_header  = l_hierarchy_header1
            changing
            it_outtab           = fil_tabstr."table must be empty

*Event receiver decl for Tree.
  data: tree_event_receiver1 type ref to cl_tree_event_receiver1.
*Event recvr for tool bar.
*The method inside this class is for class cl_gui_object . This class
*is subclass of cl_gui_object which is also a super class for
*cl_gui_alv_tree.This is how these two classes are linked
  data: l_event_receiver type ref to lcl_toolbar_event_recv.
*Before using the events we need to register those events first.
  clear l_event.
  l_event-eventid = cl_gui_column_tree=>eventid_expand_no_children.
*cl_tree_control_base
  append l_event to lt_events1.

*Event recver object for tree.
```

© 2006 SAP AG   The SAP Developer Network: http://sdn.sap.com

```
*The event recvr is created as an object for the class which implements
*the methods for ALV tree.
  create object tree_event_receiver1.
* register events
  call method g_alv_tree1->set_registered_events
    exporting
      events                  = lt_events1
    exceptions
      cntl_error               = 1
      cntl_system_error         = 2
      illegal_event_combination = 3.
  if sy-subrc <> 0.

    message e003(zmc).
  endif.
* set handler for tree1 events
*Set handler event recvr object---> method name for alv tree object name
*this is how event recvr and alv tree methods are linked.
  set handler tree_event_receiver1->handle_expand_apnc for g_alv_tree1.
  set handler tree_event_receiver1->handle_on_drop for g_alv_tree1.
  perform app_folder_list.
* §5. Send data to frontend.
  call method g_alv_tree1->frontend_update.
  v_apinit = '2'.

endform.                    " tree_initial1
*&---------------------------------------------------------------------*
*&      Form  app_folder_list
*&---------------------------------------------------------------------*
form app_folder_list.

  data: p_fipa(100) type c.

*Now start creating the hierarchy and filling the table.
  if ( ( fil_aptab[] is initial ) and  ( dir_aptab[] is initial ) ).

    if not appserv is initial.
      condense: appserv no-gaps.
      p_fipa+0(7) = 'ls -al'.
      p_fipa+7(50) = appserv.
      translate p_fipa to lower case.
      refresh outtabl.
      call 'SYSTEM' id 'COMMAND' field p_fipa
                    id 'TAB'     field outtabl-*sys*.
      sort outtabl.

    endif.
    clear: dir_aptab[],dir_aptab,fil_aptab[],fil_aptab.
    loop at outtabl.
      if ( outtabl+0(1) = 'd' ) or ( outtabl+0(1) = 's' ) or
      ( outtabl+0(1) = 'l' )  or ( outtabl+0(1) = 'D' ).
        append outtabl to dir_aptab.
      else.
        append outtabl to fil_aptab.
      endif.
    endloop.

*    commit work.
    wait up to 2 seconds.

    if  fil_aptab[] is initial.
      if  dir_aptab[] is initial .
*         message e001(zmc).
      endif.
      write: 'go'.
    endif.

  else.
    if not appserv is initial.
      condense: appserv no-gaps.
      p_fipa+0(7) = 'ls -al'.
      p_fipa+7(50) = appserv.
      translate p_fipa to lower case.
      refresh outtabl.
      call 'SYSTEM' id 'COMMAND' field p_fipa
                    id 'TAB'     field outtabl-*sys*.
```

© 2006 SAP AG

```
    sort outtabl.
  endif.
  sort outtabl.

  clear: dir_aptab1[],dir_aptab1,fil_aptab1[],fil_aptab1.
  loop at outtabl.
    if ( outtabl+0(1) = 'd' ) or ( outtabl+0(1) = 's' ) or
    ( outtabl+0(1) = 'l' ) or  ( outtabl+0(1) = 'D' ) .
      append outtabl to dir_aptab1.
    else.
      append outtabl to fil_aptab1.
    endif.
  endloop.

endif.
*    commit work.
wait up to 2 seconds.

if ( ( fil_aptab1[] is initial ) and ( dir_aptab1[] is initial )
  and ( v_apinit <> '2' ) ).
  fil_aptab1[] = fil_aptab[].
  dir_aptab1[] = dir_aptab[].
  append lines of dir_aptab to fil_aptab.
  clear v_apinit.
endif.
describe table fil_aptab1 lines no_apfile.
append lines of dir_aptab1 to fil_aptab1.

v_node_num = v_node_apnum.
if v_apinit is initial.

  perform add_this_as_apleaf1 using appserv p_relat_key changing
  v_node_num .
  v_apinit = '1'.
endif.

loop at fil_aptab1.
```

© 2006 SAP AG The SAP Developer Network: http://sdn.sap.com

```
    move fil_aptab1-pathname+54(100) to v_apfile.

    if fil_aptab1-pathname+0(1) = 'l'.
     search fil_aptab1 for '->' starting at sy-tabix ending at sy-tabix.
      sy-fdpos = sy-fdpos - 54.
      move fil_aptab1-pathname+54(sy-fdpos) to v_apfile.
    endif.

    if v_apinit ne '2'.
      perform add_this_as_apleaf using v_apfile changing v_node_num
          v_node_num1 .
    else.
      if v_apincr_node is initial.
        v_node_num = v_node_num + 1.
        v_apincr_node = 'X'.
      endif.
      perform add_this_as_apleafn using v_apfile  v_node_num changing
             v_node_num1 .
    endif.
  endloop.

endform.


*Form to add nodes to app serv container
form add_this_as_apleaf1 using    p_appserv  p_relat_key
                      changing p_v_node_num .

  data: l_node_layout1 type lvc_s_layn.

  data: l_node_text1 type lvc_value.
  l_node_text1 = p_appserv.

  if sy-tabix gt no_apfile.
    l_node_layout1-expander  = 'X'.
  endif.

  call method g_alv_tree1->add_node
```

© 2006 SAP AG   The SAP Developer Network: http://sdn.sap.com

```
    exporting
          i_relat_node_key = p_relat_key
          i_relationship   = cl_gui_column_tree=>relat_first_child
          i_node_text      = l_node_text1
         is_outtab_line = fil_tab-pathname
         is_node_layout = l_node_layout1
          importing
          e_new_node_key = p_v_node_num.

  clear l_node_layout1.
endform.                    " add_this_as_leaf

*&---------------------------------------------------------
*Form to add nodes from root/first parent
*&---------------------------------------------------------

form add_this_as_apleaf using    p_v_apfile  changing
p_v_node_num p_v_node_num1 .

  data: l_node_layout1 type lvc_s_layn.

  data: l_node_text1 type lvc_value.
  l_node_text1 = p_v_apfile.


  if sy-tabix gt no_apfile.
    l_node_layout1-expander  = 'X'.
  endif.
  l_node_layout1-dragdropid = g_handle_alv.

  call method g_alv_tree1->add_node
    exporting
          i_relat_node_key = p_v_node_num
          i_relationship   = cl_gui_column_tree=>relat_first_child
          i_node_text      = l_node_text1
         is_node_layout = l_node_layout1
          importing
          e_new_node_key = p_v_node_num1.
```

```
*To be changed.
*******************************************************
  if sy-tabix gt no_apfile.
    wa_node_det-name = p_v_apfile.
    wa_node_det-node_num = p_v_node_num1.
    wa_node_det-par_name = appserv.
    append wa_node_det to node_apdet.
  endif.

  wa_node_det-name = p_v_apfile.
  wa_node_det-node_num = p_v_node_num1.
  wa_node_det-par_name = appserv.
  append wa_node_det to fil_aptab2.
*******************************************************

  clear l_node_layout1.
  if  v_apinit eq '1'.
    p_v_node_num = 1.
  else.
    p_v_node_num = v_node_num.
  endif.

  clear l_node_layout1.

endform.

*&---------------------------------------------------------
*Form to add nodes from root/first parent1
*&---------------------------------------------------------
form add_this_as_apleafn using    p_v_apfile
p_v_node_num  changing p_v_node_num1 .

  data: l_node_layout1 type lvc_s_layn.

  data: l_node_text1 type lvc_value.
  l_node_text1 = p_v_apfile.
```

© 2006 SAP AG

```
  if sy-tabix gt no_apfile.
    l_node_layout1-expander  = 'X'.
  endif.

  l_node_layout1-dragdropid = g_handle_alv.

  call method g_alv_tree1->add_node
      exporting
            i_relat_node_key = p_v_node_num
         i_relationship   = cl_gui_column_tree=>relat_last_child
*           is_outtab_line = p_v_apfile
          is_outtab_line = appserv
            i_node_text      = l_node_text1
           is_node_layout = l_node_layout1
            importing
            e_new_node_key = p_v_node_num1.


*To be modified
**************************************************
  if sy-tabix gt no_apfile.
    wa_node_det-name = p_v_apfile.
    wa_node_det-node_num = p_v_node_num1.
    wa_node_det-par_name = appserv.
    append wa_node_det to node_apdet.
  endif.

  wa_node_det-name = p_v_apfile.
  wa_node_det-node_num = p_v_node_num1.
  wa_node_det-par_name = appserv.
  append wa_node_det to fil_aptab2.
**************************************************

  clear l_node_layout1.

endform.                    " add_this_as_leaf
```

```
*&---------------------------------------------------------------------*
*&      Form  fetch_apfile_name
*&---------------------------------------------------------------------*
form fetch_apfile_name.

  describe table fil_aptab lines tot_aprec .

  if v_node_apnum le tot_aprec.
    read table fil_aptab index v_node_apnum.
    concatenate p_apath1 '/' fil_aptab-pathname+54(100)  into appserv.
  else.
    v_node_apnum = v_node_apnum + 1.
    read table node_apdet with key node_num =  v_node_apnum .
    concatenate node_apdet-par_name '/'  node_apdet-name  into appserv.
    v_node_apnum = v_node_apnum - 1.
  endif.
  perform app_folder_list.
endform.                    " fetch_file_name
*&---------------------------------------------------------------------*
*&      Form  init_dragdrop
*&---------------------------------------------------------------------*
form init_dragdrop.
* set allowed drop effect
  g_dropeffect = cl_dragdrop=>move.
* Initialize drag & drop descriptions
* -> tree
  create object dragdrop_tree1.
  effect = cl_dragdrop=>move + cl_dragdrop=>copy.

  call method dragdrop_tree1->add
  exporting
                              flavor = 'LINE'
                              dragsrc = 'X'
                              droptarget = ''
                              effect = cl_dragdrop=>copy
.
  call method dragdrop_tree1->get_handle importing
                                       handle = g_handle_tree.
```

```
* -> ALV grid
  create object dragdrop_tree2.

  effect = cl_dragdrop=>move + cl_dragdrop=>copy.

  call method dragdrop_tree2->add
  exporting
                          flavor = 'LINE'
                          dragsrc = ''
                          droptarget = 'X'
                          effect = cl_dragdrop=>copy
.
  call method dragdrop_tree2->get_handle importing
                                      handle = g_handle_alv.
endform.                    " init_dragdrop
```

Author Bio

Prabaharan Gnanasekaran is working with Wipro Technologies as an SAP Technical Consultant for past 18 months.

Disclaimer & 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.