

# Web Dynpro for ABAP Tips and Tricks / Best Practices

---

27.01.2010

## Contents

Introduction .....	2
Summary .....	2
Web Dynpro for ABAP.....	2
Prerequisites .....	2
Tips and tricks .....	3
Where to store business logic?.....	3
How to simplify context reading/writing? .....	4
Use structure vs fields.....	5
Handle DropdownByKey .....	6
How to call Smartforms from Web Dynpro? .....	8
How to call Adobe Forms from Web Dynpro? .....	10
Appendix .....	12
The helper class.....	12
Software components used .....	18
About the developers .....	18
References .....	18
Keywords.....	18

## Introduction

### Summary

This is a collection of tips and tricks and some best practices we encountered in our projects using Web Dynpro for ABAP (WD4A). My goal is not to explain theoretic background, but to give you easy to use solutions you can spare time with and thus deliver better functionality to your clients.

A few years passed by since the 2006 publishing of “Web Dynpro for ABAP” (Praxisbuch Web Dynpro for ABAP) from U.Hoffman. As clients really started to use this new development environment there is room for new experience to be shared.

### Web Dynpro for ABAP

This development environment is abbreviated often like WD4A, but WDA, WDF4A or WD for ABAP can be seen as well. According to Wikipedia, it is a [proprietary web application user interface technology developed by SAP AG](#). It exists in a Java (Web Dynpro for Java, WDJ or WD4J) and an ABAP flavor. Basically you can develop internet or intranet applications using an IDE fully integrated into the standard SE80 Object Navigator. In this sense it is similar to Business Server Pages(BSP), but it is much more robust and has a bigger collection of standard UI elements.

### Prerequisites

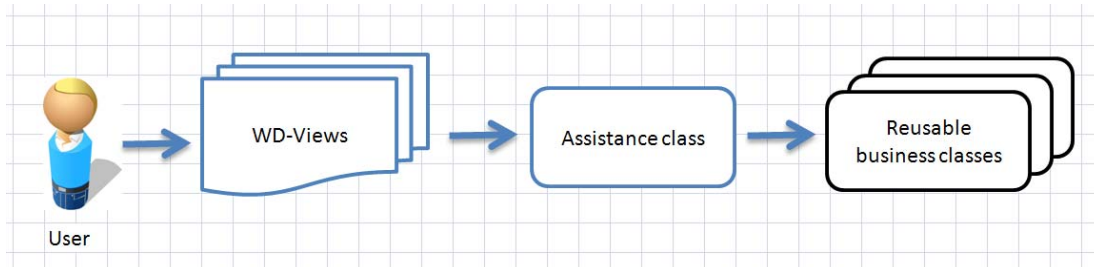
To clearly understand the following text you will need

1. Good understanding of Object Oriented ABAP
2. Basic Web Dynpro concepts like context or hook methods
3. Experience in WD4A development

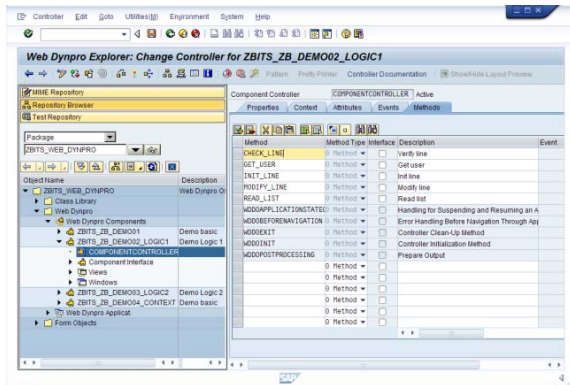
## Tips and tricks

### Where to store business logic?

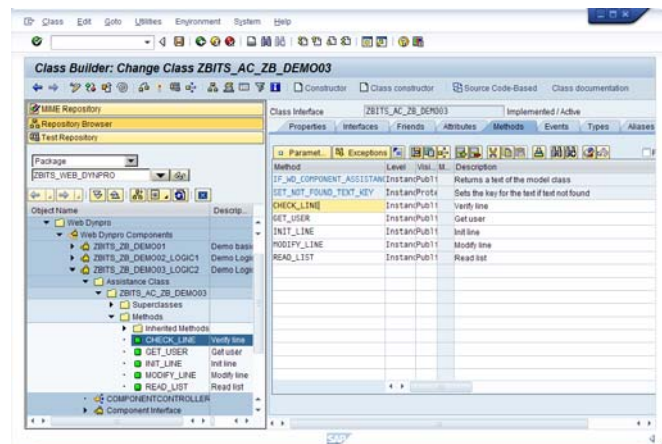
As a beginner I stored quite much logic in my main component controller. Now I try to separate reusable logic away from the component to assistance class and to other business classes as seen on the following illustration.



This way you can use business routines again or even change/copy assistance class for another purpose.



Instead of deploying your methods here...



...you put them into the assistance class.

This way you have a good overview in the left browser.

For more information on code "architecture" read WD4A from U.Hoffmann chapter "Model of Layer Separation" p185.

## How to simplify context reading/writing?

**About 60% or more of software costs is maintenance** (Boehm B. "Software Engineering Economics").

The more code lines you have, the more maintenance costs appears in your budget. So it makes sense to minimize the number of code lines. SAP standard developers seemed to ignore this fact by developing context access for WD4A. This is the standard way using Web Dynpro Code Wizard to generate code:

```
method DEMO_READ_CONTEXT1 .

    DATA lo_nd_user TYPE REF TO if_wd_context_node.
    DATA lo_nd_usr01 TYPE REF TO if_wd_context_node.
    DATA lo_el_usr01 TYPE REF TO if_wd_context_element.
    DATA ls_usr01 TYPE wd_this->element_usr01.
    * navigate from <CONTEXT> to <USER> via lead selection
    lo_nd_user = wd_context->get_child_node( name = wd_this->wdctx_user ).

    * navigate from <USER> to <USR01> via Lead selection
    lo_nd_usr01 = lo_nd_user->get_child_node( name = wd_this->wdctx_usr01 ).

    * get element via lead selection
    lo_el_usr01 = lo_nd_usr01->get_element( ).

    * get all declared attributes
    lo_el_usr01->get_static_attributes(
        IMPORTING
        static_attributes = ls_usr01 ).

endmethod.
```

As usually each view method is reading/writing one or more context nodes, you can have hundreds of – unnecessary – lines in your program and thousands in your system.

Instead of these 14 lines it is possible to read context in two.

```
method DEMO_READ_CONTEXT2 .

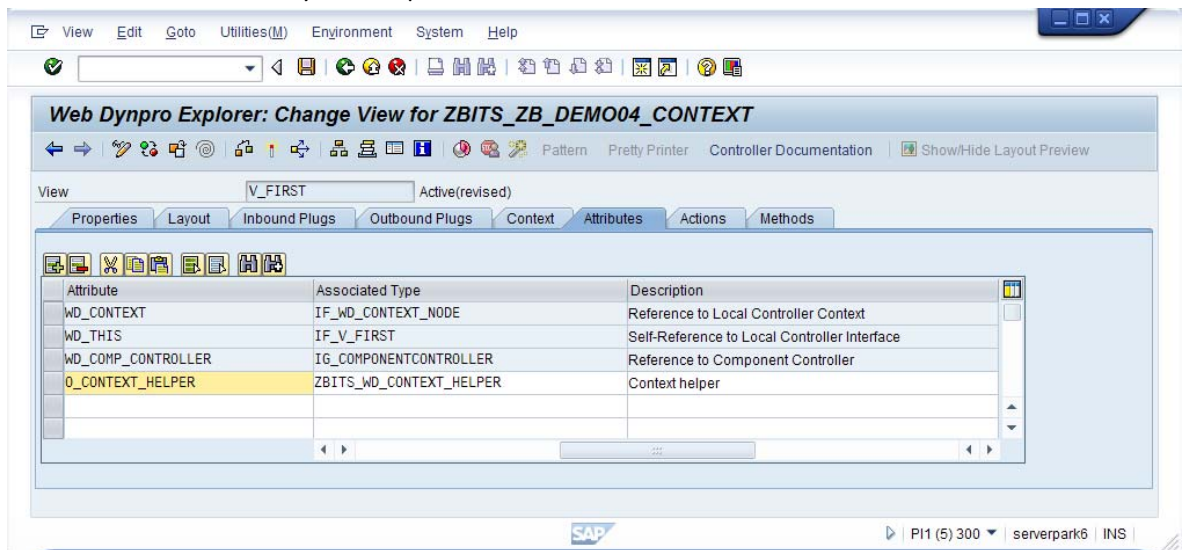
    DATA ls_usr01 TYPE wd_this->element_usr01.

    wd_this->o_context_helper-
    >get_attributes( exporting path = 'USER/USR01' importing values = ls_usr01 ).

endmethod.
```

*Other programmers did try to simplify this code clutter, but the nicest solution is from Marc Cawood. He had the idea and the first solution of integrating a helper class and using it to reach the context.*

You have a price for these simplification: you need a helper class, what you need to initialize in the WDOINIT hook method of your component:



```
method wddoinit .
    create object wd_this->o_context_helper exporting context = wd_context.
endmethod.
```

This helper class you find in the appendix.

## Use structure vs fields

This advice may not be new for you, I still find it important to mention it: Instead of using individual fields it is simpler to use complete structures.

```
method INIT_USER_FIELDS1 .
data: lv_bname type xubname,
      lv_datfm type xudatfm,
      lv_hdest type xuhdest,
      lv_menue type xumenue,
      lv_langu type xulangu.

wd_assist->init_user_fields1( exporting bname = lv_bname
                             datfm = lv_datfm
                             hdest = lv_hdest
                             menue = lv_menue
                             langu = lv_langu ).

wd_this->o_context_helper->set_attribute( exporting path = 'USER/USR01/BNAME' value = lv_bname ).
wd_this->o_context_helper->set_attribute( exporting path = 'USER/USR01/DATFM' value = lv_datfm ).
wd_this->o_context_helper->set_attribute( exporting path = 'USER/USR01/HDEST' value = lv_hdest ).
wd_this->o_context_helper->set_attribute( exporting path = 'USER/USR01/MENUE' value = lv_menue ).
wd_this->o_context_helper->set_attribute( exporting path = 'USER/USR01/LANGU' value = lv_langu ).

endmethod.
```

This way it takes 15 lines.

```
method INIT_USER_FIELDS2 .
data: ls_usr01 type usr01.
```

```

* Call assistance class to init fields
wd_assist->init_user_fields2( exporting usr01 = ls_usr01 ).

* Set context
wd_this->o_context_helper->set_attributes( exporting path = 'USER/USR01' values = ls_usr01 ).

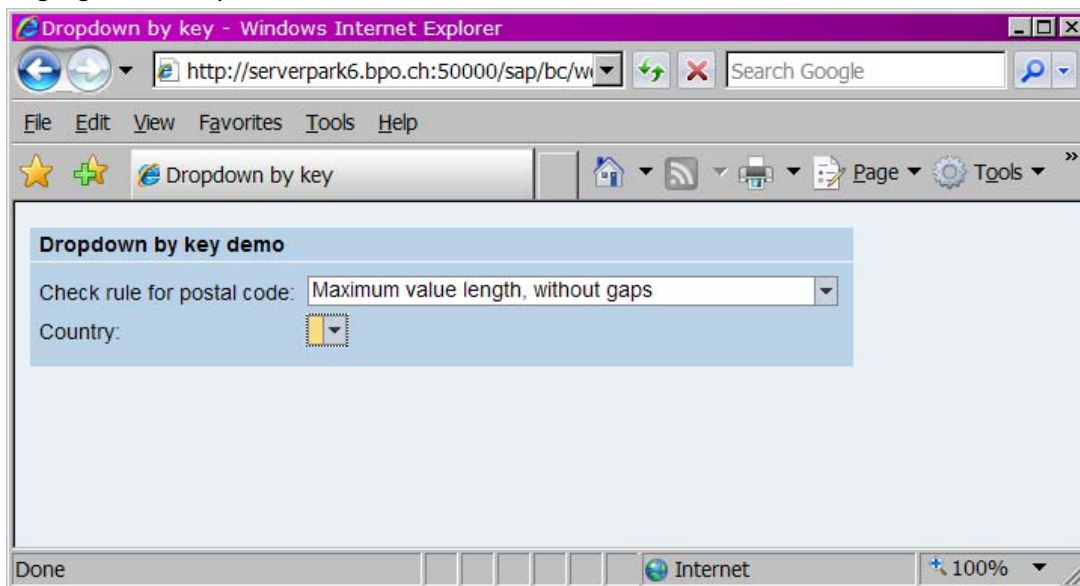
endmethod.

```

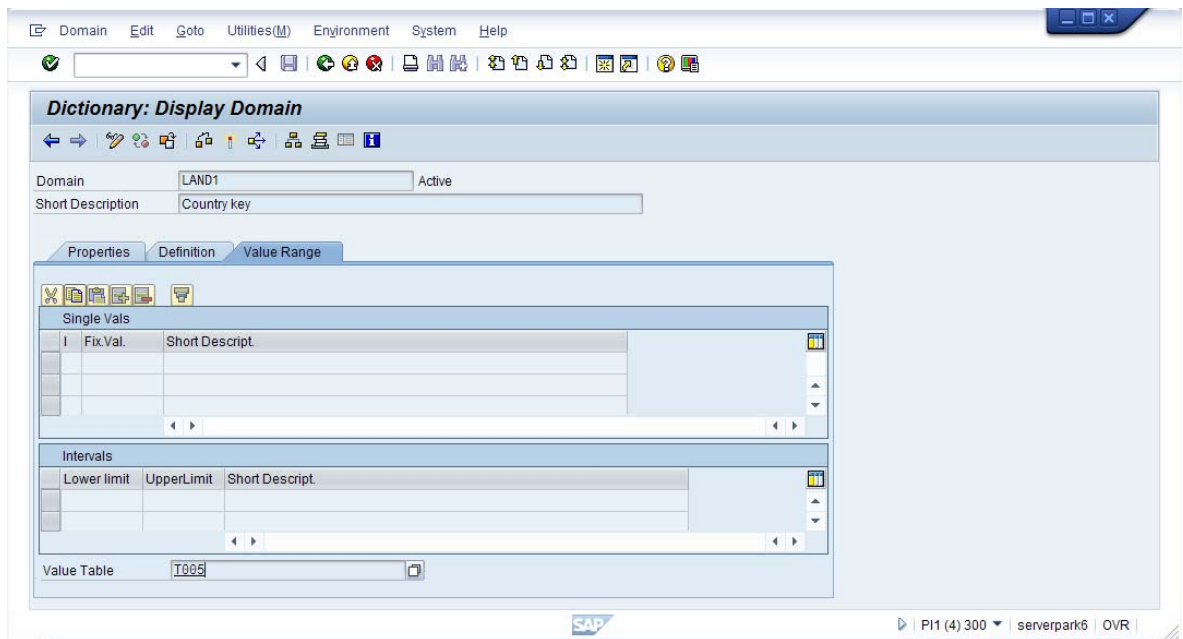
This way the same takes 3 lines.

## Handle DropdownByKey

Dropdown list are not advised by usability experts – for example by Jakob Nielsen in “Homepage Usability” – still it is a standard and widely used tool in WD4A. You will need it for example for language or country selection.



To our great surprise the WD4A engine is filling dropdown by key for certain domains, for others not. For domains, that have direct values assigned everything is filled automatically. If a domain has values indirectly through a table, no values are filled.



So if you have a dropdown by key field with this domain, no values will be filled automatically. You have to call well hidden SET\_ATTRIBUTE\_VALUE\_SET of IF\_WD\_CONTEXT\_NODE\_INFO, what you find by calling GET\_NODE\_INFO of the node object.

```

method SET_COUNTRY_VALUES1.

    data: lo_node_info type ref to if_wd_context_node_info.
    data: lo_nd_t005 type ref to if_wd_context_node.
    data: ls_t005t type t005t,
          lt_t005t type table of t005t.
    data: lt_valueset type wdr_context_attr_value_list,
          l_value type wdr_context_attr_value.

    " Get node
    lo_nd_t005 = wd_context->get_child_node( name = wd_this->wdctx_t005 ).
    lo_node_info = lo_nd_t005->get_node_info( ).

    " Select country lines
    select * from t005t into table lt_t005t
           where spras = sy-langu.

    " Put country lines into value set
    loop at lt_t005t into ls_t005t.
        l_value-value = ls_t005t-land1.
        l_value-text = ls_t005t-landx.
        insert l_value into table lt_valueset.
    endloop.

    " Assign value set
    call method lo_node_info->set_attribute_value_set( exporting name = 'LAND1'
                                                         value_set = lt_valueset
                                                         ).
endmethod.

```

This code can be further optimized by a "helper" method in ZBITS\_WD\_CONTEXT\_HELPER:

```

method set_country_values2.

data: lt_valueset type wdr_context_attr_value_list,
      l_value type wdr_context_attr_value.
data: ls_t005t type t005t,
      lt_t005t type table of t005t.

" Select country lines
select * from t005t into table lt_t005t
      where spras = sy-langu.

" Put country lines into value set
loop at lt_t005t into ls_t005t.
  l_value-value = ls_t005t-land1.
  l_value-text = ls_t005t-landx.
  insert l_value into table lt_valueset.
endloop.

" Assign value set
wd_this->o_context_helper->set_attribute_value_set( exporting path = 'T005'
                                                    name = 'LAND1'
                                                    value_set = lt_valueset ).

endmethod.

```

## How to call Smartforms from Web Dynpro?

Adobe Forms is the new technology, but sometimes it is not implemented yet or you just have the an existing form in Smartforms. This is why I prepared a simple example to call a Smartforms in WD4A. You will have a button called "Print Form" or something like that pointing onto this method. The calling action/method looks like this:

```

method ONACTIONSHOW_SMARTFORM .

data : l_pdfstring type xstring.

" Select for print
wd_assist->prepare_smartforms( importing p_pdf = l_pdfstring ).

" Call print
cl_wd_runtime_services=>attach_file_to_response(
i_filename      = 'smartform.pdf'
i_content       = l_pdfstring
i_mime_type     = 'application/pdf'
i_in_new_window = abap_false
i_inplace       = abap_false ).

endmethod.

```

There is an assistance class method behind:

```
method prepare_smartforms.
  """ S M A R T F O R M S   P D F   C A L L
  data: lv_buffer type xstring.
  data: lv_string type string.
  data: lv_fnam type rs381_fnam.
  data: lv_bytecount type i.
  data: lt_lines type table of tline.
  data: ls_line type tline.
  data: l_pdfstring type xstring.
  data: l_xline type xstring.

  data: ls_ssfctrlp type ssfctrlp.      " Control Parameters
  data: ls_output_options type ssfcompop. " Output Options
  data: ls_job_output_info type ssfcrescl. " Job Output Info
  data: ls_job_output_options type ssfcresop. " Job Output Options

  " Smartforms customer parameter
  data: ls_book type ppftbook.

  call function 'SSF_FUNCTION_MODULE_NAME'
    exporting
      formname = 'SPPFDEMO_BOOK'
    importing
      fm_name = lv_fnam.

  ls_ssfctrlp-no_dialog = 'X'.
  ls_ssfctrlp-getotf = 'X'.
  ls_ssfctrlp-preview = 'X'.

  ls_output_options-tdnoprev = 'X'.
  ls_output_options-tdtitle = sy-title.
  ls_output_options-tdnewid = 'X'.

  " Call smartforms
  call function lv_fnam
    exporting
      control_parameters = ls_ssfctrlp
      output_options     = ls_output_options
      is_book             = ls_book
    importing
      job_output_info     = ls_job_output_info
      job_output_options = ls_job_output_options
    exceptions
      formatting_error = 1
      internal_error   = 2
      send_error       = 3
      user_canceled    = 4
      others           = 5.

  " Convert to PDF
  call function 'CONVERT_OTF'
    exporting
      format = 'PDF'
    importing
      bin_filesize = lv_bytecount
  tables
    otf = ls_job_output_info-otfdata
    lines = lt_lines
```

```

exceptions
  err_conv_not_possible = 1
  err_bad_otf           = 2.

loop at lt_lines into ls_line. "into l_line.
  lv_string = ls_line.
  export my_data = lv_string to data buffer lv_buffer.
  import my_data to l_xline from data buffer lv_buffer in char-to-hex mode.

  concatenate l_pdfstring l_xline into l_pdfstring in byte mode.
endloop.

p_pdf = l_pdfstring.

endmethod. "prepare_smartforms

```

The Smartforms function is called to generate the form in OTF format. Then this OTF is converted to PDF and stored in parameter p\_pdf.

## How to call Adobe Forms from Web Dynpro?

You can find many different methods to call an adobe form. If it is enough to have a print form button, that delivers you a PDF, you can use the following two methods.

The calling action/method looks like this:

```

method ONACTIONSHOW_ADOBE_FORM .

  data : l_pdfstring type xstring.

  " Select for print
  wd_assist->prepare_adobe_forms( importing p_pdf = l_pdfstring ).

  " Call print
  cl_wd_runtime_services=>attach_file_to_response(
  i_filename      = 'adobe_form.pdf'
  i_content       = l_pdfstring
  i_mime_type     = 'application/pdf'
  i_in_new_window = abap_false
  i_inplace       = abap_false ).

endmethod.

```

The user clicking on the print form button has to have a default printer. (Usually LOCL)

There is an assistance class method behind:

```
method prepare_adobe_forms.
*   "### ADOBE FORMS PDF CALL
data: lv_funcname type funcname.
data: ls_outputparams type sfpoutputparams.
data: ls_formoutput type fpformoutput.

call function 'FP_FUNCTION_MODULE_NAME'
  exporting
    i_name      = 'FP_TEST_DATE'
  importing
    e_funcname = lv_funcname.

ls_outputparams-nodialog = 'X'. " suppress printer dialog popup
ls_outputparams-getpdf = 'X'. " Launch print preview
call function 'FP_JOB_OPEN'
  changing
    ie_outputparams = ls_outputparams
  exceptions
    cancel          = 1
    usage_error     = 2
    system_error    = 3
    internal_error  = 4
    others          = 5.

call function lv_funcname
*   exporting
*   /1bcdwb/docparams = fp_docparams
  importing
    /1bcdwb/formoutput = ls_formoutput
  exceptions
    usage_error     = 1
    system_error    = 2
    internal_error  = 3
    others          = 4.

call function 'FP_JOB_CLOSE'
  exceptions
    usage_error     = 1
    system_error    = 2
    internal_error  = 3
    others          = 4.

p_pdf = ls_formoutput-pdf.

endmethod.                                "prepare_adobe_forms
```

## Appendix

### The helper class

You also find these parts in text files:

[http://www.zbalai.com/\\_abap/content/420\\_web\\_dynpro\\_for\\_abap\\_tips\\_and\\_tricks/HELPER.txt](http://www.zbalai.com/_abap/content/420_web_dynpro_for_abap_tips_and_tricks/HELPER.txt)

[http://www.zbalai.com/\\_abap/content/420\\_web\\_dynpro\\_for\\_abap\\_tips\\_and\\_tricks/HELPER\\_METHODS.txt](http://www.zbalai.com/_abap/content/420_web_dynpro_for_abap_tips_and_tricks/HELPER_METHODS.txt)

To easily upload the helper class you need to switch to implementation public and private section.

Public section:

```
*** public components of class ZBITS_WD_CONTEXT_HELPER
*** do not include other source files here!!!
public section.

data O_CONTEXT type ref to IF_WD_CONTEXT_NODE .

methods GET_ATTRIBUTE
  importing
    !PATH type STRING
  exporting
    value(VALUE) type DATA .
methods GET_ATTRIBUTES
  importing
    !PATH type STRING
  exporting
    !VALUES type DATA .
methods SET_ATTRIBUTE
  importing
    !PATH type STRING
    !VALUE type DATA .
methods SET_ATTRIBUTES
  importing
    !PATH type STRING
    !VALUES type DATA .
methods GET_STATIC_ATTRIBUTES_TABLE
  importing
    !PATH type STRING
    !FROM type I default 1
    !TO type I default 2147483647
  exporting
    !TABLE type ANY TABLE .
type-pools ABAP .
methods BIND_TABLE
  importing
    !PATH type STRING
    !NEW_ITEMS type ANY TABLE
    !SET_INITIAL_ELEMENTS type ABAP_BOOL default ABAP_TRUE
    !INDEX type I optional .
methods IS_CHANGED_BY_CLIENT
  importing
    !PATH type STRING
  returning
    value(FLAG) type ABAP_BOOL .
methods GET_NODE
  importing
    !PATH type STRING
  returning
    value(RESULT) type ref to IF_WD_CONTEXT_NODE .
```

```

methods SET_CONTEXT
  importing
    !CONTEXT type ref to IF_WD_CONTEXT_NODE .
methods CONSTRUCTOR
  importing
    !CONTEXT type ref to IF_WD_CONTEXT_NODE optional .
methods SET_LEAD_SELECTION
  importing
    !PATH type STRING
    !ELEMENT type ref to IF_WD_CONTEXT_ELEMENT .
methods RESET_CHANGED_BY_CLIENT
  importing
    !PATH type STRING .
methods SET_CHANGED_BY_CLIENT
  importing
    !PATH type STRING
    !FLAG type ABAP_BOOL .

```

Private section:

```

private section.

methods GET_ELEMENT_PATH
  importing
    !PATH type STRING
  exporting
    !ELPATH type STRING
    !ATTRIBUTE type DATA .
methods GET_ELEMENT
  importing
    !PATH type STRING
  returning
    value(ELEMENT) type ref to IF_WD_CONTEXT_ELEMENT .

```

Methods:

```

method bind_table.
*****
  data: lo_node          type ref to   if_wd_context_node.
*****

  " Validate necessary object initialisation
  check o_context is bound.

  " Get the element
  lo_node = get_node( path ).
  lo_node->bind_table( exporting new_items = new_items set_initial_elements =
set_initial_elements index = index ).

endmethod.

method CONSTRUCTOR.

  o_context = context.

endmethod.

method GET_ATTRIBUTE.
*****

```

```

DATA: lv_elpath      TYPE      string,
      lv_attribute   TYPE      string,
      lo_element     TYPE REF TO if_wd_context_element.
*****

" Validate necessary object initialisation
CHECK o_context IS BOUND.

" Determine element path and attribute name
get_element_path( EXPORTING path = path IMPORTING elpath = lv_elpath attribute
= lv_attribute ).

" Get the element
lo_element = get_element( lv_elpath ).
lo_element->get_attribute( EXPORTING name = lv_attribute IMPORTING value =
value ).

endmethod.

method get_attributes.
*****
data: lo_node        type ref to  if_wd_context_node.
*****

" Validate necessary object initialisation
check o_context is bound.

" Get the element
lo_node = get_node( path ).
lo_node->get_static_attributes( importing static_attributes = values ).

endmethod.

method GET_NODE.
*****
DATA: lv_path        TYPE      string,
      lt_path        TYPE TABLE OF string,
      lv_index       TYPE      i,
      lv_length      TYPE      i,
      lo_node        TYPE REF TO if_wd_context_node.
*****

" Validate necessary object initialisation
CHECK me->o_context IS BOUND.

" Read the path into a table
SPLIT path AT '/' INTO TABLE lt_path.

" Get the path depth
DESCRIBE TABLE lt_path LINES lv_length.

" Root attributes
IF lv_length = 0.
  lo_node = me->o_context.
ENDIF.

" Follow the path
lv_index = 0.
LOOP AT lt_path INTO lv_path.
  lv_index = lv_index + 1.

```

```

IF lv_index = 1.
  " Get first node in the path
  lo_node = me->o_context->get_child_node( lv_path ).
ELSE.
  " Get next node down the path
  lo_node = lo_node->get_child_node( lv_path ).
ENDIF.
ENDLOOP.

result = lo_node.

endmethod.

method get_static_attributes_table.
*****
data: lo_node      type ref to   if_wd_context_node.
*****

" Validate necessary object initialisation
check o_context is bound.

" Get the element
lo_node = get_node( path ).
lo_node->get_static_attributes_table( exporting from = from to = to importing
table = table ).

endmethod.

method IS_CHANGED_BY_CLIENT.
*****
data: lo_node      type ref to   if_wd_context_node.
*****

" Validate necessary object initialisation
check o_context is bound.

" Get the element
lo_node = get_node( path ).
flag = lo_node->IS_CHANGED_BY_CLIENT( ).

endmethod.

method RESET_CHANGED_BY_CLIENT.
*****
data: lo_node      type ref to   if_wd_context_node.
*****

" Validate necessary object initialisation
check o_context is bound.

" Get the element
lo_node = get_node( path ).
lo_node->RESET_CHANGED_BY_CLIENT( ).

endmethod.

method SET_ATTRIBUTE.
*****
DATA: lv_elpath    TYPE          string,

```

```

lv_attribute TYPE string,
lo_element TYPE REF TO if_wd_context_element.
*****

" Validate necessary object initialisation
CHECK o_context IS BOUND.

" Determine element path and attribute name
get_element_path( EXPORTING path = path IMPORTING elpath = lv_elpath attribute
= lv_attribute ).

" Get the element
lo_element = get_element( lv_elpath ).
lo_element->set_attribute( EXPORTING name = lv_attribute value = value ).

endmethod.

method set_attributes.
*****
data: lo_node type ref to if_wd_context_node.
*****

" Validate necessary object initialisation
check o_context is bound.

" Get the element
lo_node = get_node( path ).
lo_node->set_static_attributes( exporting static_attributes = values ).

endmethod.

method SET_CHANGED_BY_CLIENT.
*****
data: lo_node type ref to if_wd_context_node.
*****

" Validate necessary object initialisation
check o_context is bound.

" Get the element
lo_node = get_node( path ).
lo_node->SET_CHANGED_BY_CLIENT( flag ).

endmethod.

method SET_CONTEXT.

o_context = context.

endmethod.

method set_lead_selection.
*****
data: lo_node type ref to if_wd_context_node.
*****

" Validate necessary object initialisation
check o_context is bound.

" Get the element

```

```

lo_node = get_node( path ).
lo_node->set_lead_selection( exporting element = element ).

endmethod.

method GET_ELEMENT.
*****
DATA: lo_node      TYPE REF TO   if_wd_context_node.
*****

" Validate necessary object initialisation
CHECK me->o_context IS BOUND.

lo_node = get_node( path ).

element = lo_node->get_element( ).

endmethod.

method GET_ELEMENT_PATH.

*****
DATA: lv_path      TYPE          string,
      lt_path      TYPE TABLE OF string,
      lv_index     TYPE          i,
      lv_length    TYPE          i,
      lv_lengm1   TYPE          i.
*****

" Read the path into a table
SPLIT path AT '/' INTO TABLE lt_path.

" Get the path depth
DESCRIBE TABLE lt_path LINES lv_length.
lv_lengm1 = lv_length - 1.

" Follow the path
lv_index = 0.
LOOP AT lt_path INTO lv_path.
  lv_index = lv_index + 1.
  IF lv_index < lv_length.
    CONCATENATE elpath lv_path INTO elpath.
    IF lv_index < lv_lengm1.
      CONCATENATE elpath '/' INTO elpath.
    ENDIF.
  ELSE.
    attribute = lv_path.
  ENDIF.
ENDLOOP.

endmethod.

```

## Software components used

SAP ECC 6.0

SAP Frontend 7.10

## About the developers

Marc Cawood is an experienced SAP developer with references in WD4A field.

Zsolt Balai ([zbalai@zbalai.com](mailto:zbalai@zbalai.com)) – author of this document – is an SAP/Internet technical consultant and developer working in HR/FI/CO/MM/SD modules. He is currently employed by [Bits AG](#) and is available for projects in Switzerland.

## References

[1] Hoffman Ulli “Web Dynpro for ABAP”, SAP/Galileo Press (2006).

[2] Boehm B. “Software Engineering Economics”, Prentice Hall (1981).

[3] Jakob Nielsen & Marie Tahir “Homepage Usability” New Riders (2001).

## Keywords

WD4A, WDA, WDF4A, Web Dynpro for ABAP, tips and tricks, best practices, simple context reading