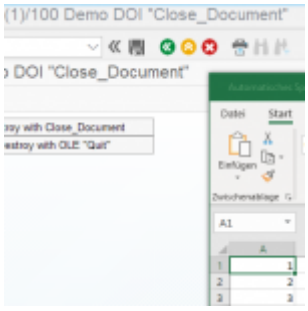


OLE-Handle zu DOI-Objekt ermitteln

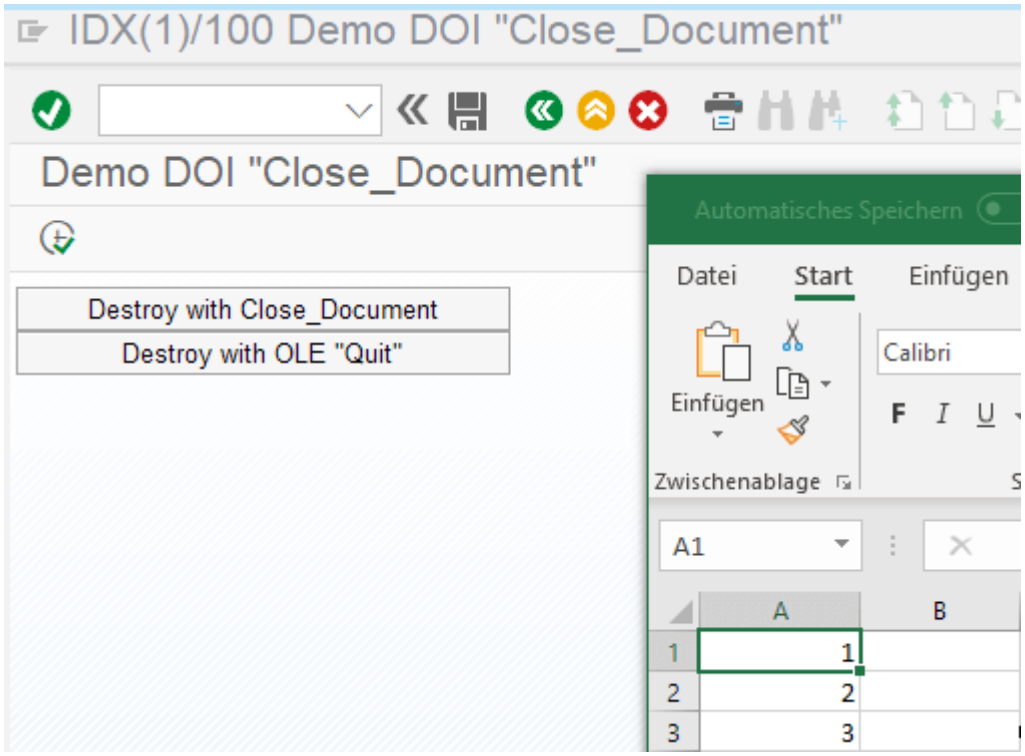
Für die Integration von Word und Excel gibt es nur zwei Möglichkeiten:

- OLE – Object Linking and Embedding
- DOI – Desktop Office Integration

Beide Varianten haben ihre Vor- und Nachteile. Ein Nachteil bei der Verwendung von DOI ist auf jeden Fall, dass die Methode CLOSE_DOCUMENT nicht zuverlässig funktioniert, wenn man Word oder Excel in einem separaten Fenster öffnet.

Beispiel

Zuerst jedoch ein Beispielcode, der zeigt, wie mittels DOI eine Excel-Instanz gestartet wird. Die Drucktaste „Create“ startet Excel.



Desktop Office Integration hat Excel gestartet

Die Schaltfläche „Create“ wird dann ausgeschaltet und die Schaltflächen „Destroy with „Close_Document“ und „Destroy with OLE Quit“ erscheinen. Mit diesen Drucktaste kann Excel wieder geschlossen werden. Das Beenden von Excel/ Word geschieht in der Regel über *document->close_document* und *document->close_activex_document*. Falls dies jedoch aus unerfindlichen Gründen nicht funktioniert, hilft vielleicht die Methode über das OLE-Objekt und die Methode „Quit“.

Code

```
REPORT LINE-SIZE 200.


"Create Control
SELECTION-SCREEN PUSHBUTTON /1(30) TEXT-cre USER-COMMAND create      MODIF ID
cre.
"Destroy Control with Close_Document
SELECTION-SCREEN PUSHBUTTON /1(30) TEXT-dst USER-COMMAND destroy     MODIF ID
dst.
"Destrpy Control with OLE and method Quit
SELECTION-SCREEN PUSHBUTTON /1(30) TEXT-dso USER-COMMAND destroy_ole MODIF ID
dst.
```

```
CLASS demo DEFINITION.
  PUBLIC SECTION.
    METHODS create.
    METHODS destroy.
    METHODS destroy_with_ole.
    METHODS is_destroyed RETURNING VALUE(result) TYPE i.
  PROTECTED SECTION.

    DATA mr_control       TYPE REF TO i_oi_container_control."
OIContainerCtrl
    DATA mr_document      TYPE REF TO i_oi_document_proxy.   " Office
Dokument
    DATA mr_spreadsheet   TYPE REF TO i_oi_spreadsheet.      " Spreadsheet

    METHODS set_data.
    METHODS create_application.

ENDCLASS.


INITIALIZATION.
  DATA(go_demo) = NEW demo( ).

AT SELECTION-SCREEN.
  CASE sy-ucomm.
    WHEN 'CREATE'.
      go_demo->create( ).
    WHEN 'DESTROY'.
      go_demo->destroy( ).
    WHEN 'DESTROY_OLE'.
      go_demo->destroy_with_ole( ).
  ENDCASE.

AT SELECTION-SCREEN OUTPUT.

  LOOP AT SCREEN.
    CASE screen-group1.
      WHEN 'DST'.
        IF go_demo->is_destroyed( ) = 0.
          screen-active = '1'.
        ELSE.
          screen-active = '0'.
        ENDIF.

      WHEN 'CRE'.
        IF go_demo->is_destroyed( ) = 0.
          screen-active = '0'.
        ELSE.
          screen-active = '1'.
        ENDIF.
    ENDCASE.
```

```
    MODIFY SCREEN.
  ENDLOOP.

CLASS demo IMPLEMENTATION.


  METHOD create.

    create_application( ).
    set_data( ).

  ENDMETHOD.

  METHOD is_destroyed.

    IF mr_document IS BOUND.
      mr_document->is_destroyed(
        IMPORTING
          ret_value = result
      ).
    ELSE.
      result = 1.
    ENDIF.

  ENDMETHOD.

  METHOD create_application.

    DATA lr_error         TYPE REF TO i_oi_error.

    c_oi_container_control_creator=>get_container_control(
        IMPORTING
          control = mr_control
          error   = lr_error ).

** init control
    mr_control->init_control(
      EXPORTING
        inplace_enabled       = abap_false
        no_flush              = 'X'
        r3_application_name   = 'Test DOI'
        inplace_show_toolbars = abap_false
        parent                = cl_gui_container=>screen9
      IMPORTING
        error                 = lr_error
      EXCEPTIONS
        OTHERS                = 2 ).

    IF lr_error->has_failed = abap_true. lr_error->raise_message( 'I' ).
RETURN. ENDIF.
```

```
*** Get Documentproxy
    CALL METHOD mr_control->get_document_proxy
      EXPORTING
        document_type  = soi_doctype_excel_sheet "'Excel.Sheet'
        no_flush       = 'X'
      IMPORTING
        document_proxy = mr_document
        error          = lr_error.
    IF lr_error->has_failed = abap_true. lr_error->raise_message( 'I' ).
RETURN. ENDIF.

    mr_document->create_document(
      EXPORTING
       document_title   = 'Demo-Arbeitsblatt'
        no_flush         = 'X'
        open_inplace     = abap_false
      IMPORTING
       error = lr_error ).
    IF lr_error->has_failed = abap_true. lr_error->raise_message( 'I' ).
RETURN. ENDIF.

    CALL METHOD mr_document->get_spreadsheet_interface
      IMPORTING
        sheet_interface = mr_spreadsheet
        error           = lr_error.
    IF lr_error->has_failed = abap_true. lr_error->raise_message( 'I' ).
RETURN. ENDIF.

  ENDMETHOD.

  METHOD set_data.

    DATA lt_values        TYPE soi_generic_table.
    DATA lt_ranges        TYPE soi_range_list.
    DATA ls_range         LIKE LINE OF lt_ranges.
    DATA lr_error         TYPE REF TO i_oi_error.


    "Demo-Daten
    lt_values = VALUE #(
        ( row = 1 column = 1 value = '1' )
        ( row = 2 column = 1 value = '2' )
        ( row = 3 column = 1 value = '3' ) ) .

*== Neuen Bereich definieren
    mr_spreadsheet->insert_range_dim(
       EXPORTING name = 'myarea'
                 top       = 1
                 left      = 1
                 rows      = lines( lt_values )
                 columns   = 1
```

```
                no_flush  = abap_false ).
    ls_range-columns = 1.
    ls_range-rows    = lines( lt_values ).
    ls_range-name    = 'myarea'.
    APPEND ls_range TO lt_ranges.

    "Daten übergeben
    mr_spreadsheet->set_ranges_data(
      EXPORTING
        ranges    = lt_ranges
        contents  = lt_values
      IMPORTING
        error     = lr_error
    ).
    IF lr_error->has_failed = abap_true. lr_error->raise_message( 'I' ).
RETURN. ENDIF.

  ENDMETHOD.

  METHOD destroy.

    mr_document->close_document( ).
    mr_document->close_activex_document( ).
    FREE mr_document.

    mr_control->destroy_control( ).
    FREE mr_control.

  ENDMETHOD.

  METHOD destroy_with_ole.

    DATA lv_document_cntl_handle TYPE cntl_handle.
    DATA lv_application          TYPE ole2_object.
    DATA lv_oi_ret               TYPE soi_ret_string.
    DATA lr_error                TYPE REF TO i_oi_error.

    mr_document->get_document_handle(
      IMPORTING
        handle  = lv_document_cntl_handle
        retcode = lv_oi_ret ).
    GET PROPERTY OF lv_document_cntl_handle-obj 'Application' =
lv_application.
    CALL METHOD OF lv_application 'Quit'.
    FREE OBJECT lv_application.
    FREE mr_document.

  ENDMETHOD.

ENDCLASS.
```

OLE-Objekt zu DOI-Dokument erhalten

Wenn die Methode *close_document* aus irgendwelchen Gründen nicht funktioniert, das Excel-Fenster also nicht geschlossen wird, oder du eine Methode anwenden möchtest, die das DOI-Interface nicht unterstützt, dann kannst du dir den OLE-Handle zur Applikation besorgen:

```
    DATA lv_document_cntl_handle TYPE cntl_handle.
    DATA lv_application          TYPE ole2_object.
    DATA lv_oi_ret               TYPE soi_ret_string.
    DATA lr_error                TYPE REF TO i_oi_error.

    mr_document->get_document_handle(
      IMPORTING
        handle  = lv_document_cntl_handle
        retcode = lv_oi_ret ).
    GET PROPERTY OF lv_document_cntl_handle-obj 'Application' =
lv_application.
    CALL METHOD OF lv_application 'Quit'.
    FREE OBJECT lv_application.
    FREE mr_document.
```

SAP-Demoprogramme

Folgende zwei Demoprogramme zeigen noch erweiterte Funktionen des DOI-Interfaces:

- SAPRDEMO_FORM_INTERFACE
- SAPRDEMO_SPREADSHEET_INTERFACE
- SAPRDEMO_MAILMERGE_INTERFACE

Bei diesen drei Programmen lässt sich übrigens der Fehler, dass die Applikation nicht korrekt geschlossen wird, sehr gut nachstellen.

Bei SAPRDEMO_MAILMERGE_INTERFACE wird beim Beenden des Programms zwar das Dokument geschlossen, aber die Word-Applikation bleibt bestehen. Bei SAPRDEMO_FORM_INTERFACE wird das Programm zwar beendet, doch obwohl Close_Document und Destroy_Control aufgerufen werden, bleibt das Excel-Fenster geöffnet. Im Programm SAPRDEMO_SPREADSHEET_INTERFACE funktioniert das Beenden von Excel wiederum perfekt.