

Automatisch generierte Listereports mit CDS Views und ALV

rku IT | DevGarage



www.rku-it.de

1. Ausgangslage
2. Zeitliche Entwicklung
3. Datenmodell für die Demonstration
4. Auswertung von CDS-Annotationen im eigenen Coding
5. Aufbau des ALV-Grids aus CDS-Annotationen
6. Event Handling mit Intent-Based Navigation und BOPF-Aktionen
7. Selektion mit dem SADL-Framework
8. Generierung von Reports
9. Motivation und Nutzen

Entwicklung von Listenreports

In der Praxis ergeben sich oft Anforderungen, Daten aus der Datenbank zu selektieren und als Liste anzuzeigen. Hierbei ergeben sich oft wiederkehrende Schritte in der Entwicklung:

- Anlegen eines Selektionsbilds
- Selektion der Daten aus einer oder mehreren Datenbanktabellen
- Aufbau eines ALV-Grids mit Feldkatalog, Layout und Funktionen
- Event Handling für Hotspots und User Commands

Hierbei entsteht viel Boilerplate Code, der sich regelmäßig wiederholt, woraus sich die Motivation ergibt, diesen in eine wiederverwendbares Framework auszulagern. CDS Views und ihre Annotationen bieten eine Grundlage für ein solches Framework.

Facade-Klasse zum Aufbau von ALV-Grids

- **2017 – 2018:** Entwicklung einer Reihe von Reports im Rahmen der Umsetzung des Interimsmodell für die Marktkommunikation in der Versorgungsbranche. Identifikation wiederkehrender Anforderungen und Beginn der Entwicklung einer Facade-Klasse für einen effizienten Aufbau von ALV-Grids (analog zu „ABAP to the Future“)
- **2018 – 2020:** Weiterentwicklung der Facade-Klasse und Erweiterung um neue Funktionen, jeweils getrieben durch Anforderungen aus aktuellen Projekten und dem Tagesgeschäft.

Framework für ALV-Reports für CDS-Views (1)

- **2020:** Entwicklung einer Klasse zur automatischen Versorgung der Facade-Klasse mit Informationen aus den UI-Annotationen eines CDS Views und einer Klasse zur Erfassung freier Selektionen auf Basis der Consumption-Annotationen des CDS Views.
- **2021:** Entwicklung der automatischen Generierung von Selektionsbildern anhand der Consumption- und UI-Annotationen, sowie des Split Screen Views; Anbindung von SADL und BOPF für Selektion, ändernde Zugriffe, und Aktionen analog zu den generierten OData Services; Realisierung des Event Handlings über Intent Based Navigation und BOPF-Aktionen

Framework für ALV-Reports für CDS-Views (2)

- **2021:** Neue Generierungsstrategie mit der Möglichkeit zur Einbindung von Add-Ins, Beginn der Nutzung des Virtual Data Models und Rollout der ersten Transaktionen auf Basis des Frameworks und des VDM (Skalierung).

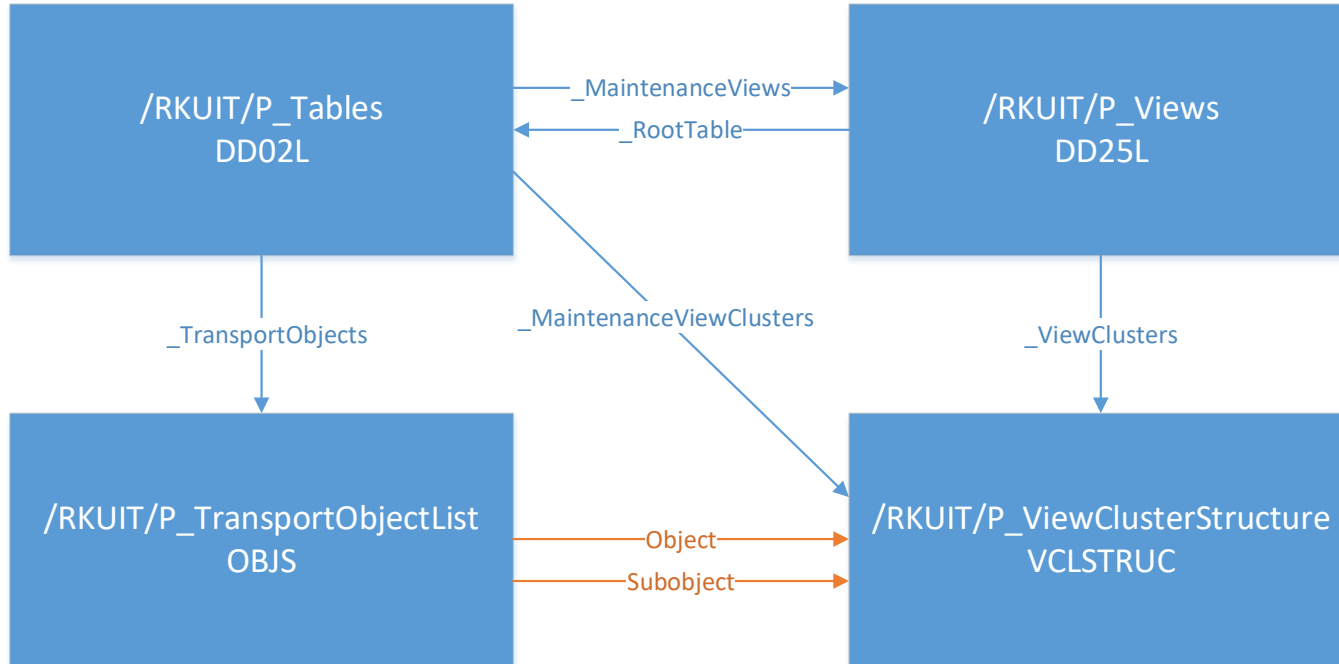
Erinnerung

Das Virtual Data Model ist eine hierarchische Struktur, die CDS Views nach Komplexität und Nutzungszweck aufteilt:

- **Interface Views:** sind zur Wiederverwendung gedacht und können weiter unterteilt werden:
 - **Basic Views:** exponieren eine Datenbanktabelle als Quelle, sowie ihre Fremdschlüsselbeziehungen als Assoziationen.
 - **Composite Views:** exponieren mehrere Datenbanktabellen, die über Joins oder Pfadausdrücke verknüpft sind.
 - **Transactional Views:** exponieren eine Datenbanktabelle für ändernde Zugriffe
- **Consumption Views:** werden genutzt, um die Daten aus einem oder mehreren Interface Views einer Anwendung zur Verfügung zu stellen und diese mit Informationen zur Verarbeitung und Anzeige anzureichern

Datenmodell für die Demonstration

Pflegeobjekte zu einer Datenbanktabelle (Basic-Views, Assoziationen)



Datenmodell für die Demonstration

Pflegeobjekte zu einer Datenbanktabelle (Weitere Views)

- [Composite Interface View](#)
- [Consumption View](#)

Demonstration (1): Report ohne Frameworkkomponenten

[\[DEMO\]](#)

Auswertung von CDS-Annotationen im eigenen Coding

Ermittlung aller Annotationen eines CDS-Views als Name-Wert-Paare

```
METHOD get_annotations_for_cds_view.  
  cl_dd_ddl_annotation_service=>get_annos(  
    EXPORTING  
      entityname      = to_upper( iv_cds_view )  
    IMPORTING  
      entity_annos    = et_entity_annotations  
      element_annos   = et_element_annotations  
      parameter_annos = et_parameter_annotations ).  
ENDMETHOD.
```

Auswertung von Anotationen für die Anzeige

```
WHEN 'CONSUMPTION.HIDDEN'.
    INSERT lv_fieldname INTO TABLE at_technical.

WHEN 'DEFAULTAGGREGATION'.
    INSERT VALUE #(
        fieldname = lv_fieldname
        aggregation = SWITCH #( <element_annotation>-value
            WHEN '#SUM' THEN if_salv_c_aggregation=>total
            WHEN '#MAX' THEN if_salv_c_aggregation=>maximum
            WHEN '#MIN' THEN if_salv_c_aggregation=>minimum
            WHEN '#AVG' THEN if_salv_c_aggregation=>average
            ELSE if_salv_c_aggregation=>none ) )
    INTO TABLE at_aggregations.

WHEN 'UI.HIDDEN'.
    INSERT lv_fieldname INTO TABLE at_hidden.
```

Übergabe der Informationen an die Facade-Klasse

```
io_view->setup_table(  
    iv_set_functions      = iv_set_functions  
    it_add_functions      = lt_add_functions  
    io_top_of_list        = lo_top_of_list  
    io_event_handler      = lo_event_handler  
    iv_setup_columns      = abap_true  
    it_keys               = lt_keys  
    it_technical          = lt_technical  
    it_hidden             = lt_hidden  
    it_hotspots           = lt_hotspots  
    it_buttons            = lt_buttons  
    it_checkboxes         = lt_checkboxes  
    it_editable           = lt_editable  
    iv_style_fieldname    = iv_style_fieldname  
    iv_start_in_edit_mode = iv_start_in_edit_mode  
    it_fielddtexts        = lt_fielddtexts  
    it_conversion         = lt_conversion  
    it_ordering           = lt_ordering  
    it_aggregations       = lt_aggregations ).
```

Wirkung einzelner Annotationen (Auszug)

Annotation	Wirkung
UI.hidden	Die Spalte wird ausgeblendet
UI.lineItem.position	positioniert die Spalte im ALV-Grid, intern werden alle Lücken bei der Positionierung geschlossen, da die Anzeige sonst nicht korrekt funktioniert
EndUserText.label	Setzt die Überschrift der Spalte
EndUserText.quickinfo	Setzt den Tooltip der Spalte
DefaultAggregation	Setzt die initiale Aggregation der Daten

Aufbau des ALV-Grids aus den CDS-Annotationen

Demonstration (2): Report mit Feldkatalog aus Assoziationen

[\[DEMO\]](#)

Auswertung der Annotation UI.lineitem (1)

```
CASE ls_ui_anno-key.  
  WHEN 'UI.LINEITEM.SEMANTICOBJECTACTION'.  
    ls_field_action-semantic_action = replace( val = ls_ui_anno-value sub = `` with = `` occ = 0 ).  
  
  WHEN 'UI.LINEITEM.DATAACTION'.  
    ls_field_action-data_action = replace( val = ls_ui_anno-value sub = `` with = `` occ = 0 ).  
  
  WHEN 'UI.LINEITEM.TARGETELEMENT'.  
    ls_field_action-associationname = replace( val = ls_ui_anno-value sub = `` with = `` occ = 0 ).  
  
  WHEN 'UI.LINEITEM.URL'.  
    ls_field_action-url_fieldname = to_upper( replace( val = ls_ui_anno-value sub = `` with = `` occ = 0 ) ).  
  
  WHEN 'UI.LINEITEM.TYPE'.  
    ls_field_action-fieldtype = ls_ui_anno-value.  
  
ENDCASE.
```

Auswertung der Annotation UI.lineItem (2)

```
WHEN '#WITH_INTENT_BASED_NAVIGATION'.  
  " für den Typen WITH_INTENT_BASED_NAVIGATION wird das Feld zum Hyperlink  
  IF ls_field_action-semantic_object IS NOT INITIAL AND  
    ls_field_action-semantic_action IS NOT INITIAL.  
    ls_field_action-hotspot = abap_true.  
    ls_field_action-action_type = ac_action_type-hotspot.  
  
    INSERT lv_fieldname INTO TABLE at_hotspots.  
    INSERT ls_field_action INTO TABLE at_field_actions.  
  ENDIF.
```

Auswertung der Annotation UI.lineItem (3)

```
WHEN '#FOR_INTENT_BASED_NAVIGATION'.
```

```
" Für den Typen FOR_INTENT_BASED_NAVIGATION wird ein Button angelegt
```

```
IF ls_field_action-semantic_object IS NOT INITIAL AND
```

```
   ls_field_action-semantic_action IS NOT INITIAL.
```

```
lv_action_counter = lv_action_counter + 1.
```

```
ls_field_action-user_command = |{ ac_function_code_prefix }{ lv_action_counter }|.
```

```
ls_field_action-action_type = ac_action_type-intent_based_navigation.
```

```
INSERT ls_field_action INTO TABLE at_field_actions.
```

```
INSERT VALUE #(
```

```
    name      = ls_field_action-user_command
```

```
    text      = COND #( WHEN lv_field_label IS NOT INITIAL THEN lv_field_label  
                        ELSE |{ ls_field_action-semantic_object }| && '.' &&  
                        |{ ls_field_action-semantic_action }| )
```

```
    position = if_salv_c_function_position=>right_of_salv_functions )
```

```
INTO TABLE at_add_functions.
```

```
ENDIF.
```

Wirkung einzelner Annotationen (Auszug)

Annotation	Wirkung
UI.lineItem.type	<p>Der Wert bestimmt, wie der Aufruf von Funktion für das Feld umgesetzt wird:</p> <ul style="list-style-type: none">• #FOR_ACTION: In der Toolbar wird ein Button für eine BOPF-Aktion angelegt• #FOR_INTENT_BASED_NAVIGATION, #WITH_NAVIGATION_PATH: In der Toolbar wird ein Button für eine Navigation angelegt• #WITH_INTENT_BASED_NAVIGATION, #WITH_URL: Die Spalte wird Hotspot

Wirkung einzelner Annotationen (Auszug)

Annotation	Wirkung
UI.lineItem.targetElement	Spezifiziert die Assoziation für die Navigation (#WITH_NAVIGATION_PATH)
UI.lineItem.semanticObjectAction	Spezifiziert die Aktion für Intent-Based Navigation (#FOR_INTENT_BASED_NAVIGATION, #WITH_INTENT_BASED_NAVIGATION)
UI.lineItem.url	Spezifiziert ein Viewfeld, dass die Ziel-URL für den Hotspot-Klick enthält (#WITH_URL)
UI.lineItem.dataAction	Spezifiziert eine mit dem Feld verknüpfte BOPF-Aktion (# FOR_ACTION)

Verfügbare Funktionen

1. Auffrischen, Speichern, Löschen, Anzeigen/Ändern
2. Intent-Based Navigation
3. Navigation entlang einer Assoziation
4. BOPF-Aktion
5. URL aufrufen
6. Mail versenden
7. Eigene Aktion (muss durch Unterklasse redefiniert werden)

Aufruf der Navigation (Intent-Based oder via Assoziation)

```
IF ls_action-semantic_object IS NOT INITIAL AND
   ls_action-semantic_action IS NOT INITIAL.
  " Variante 1: Intent-based Navigation
  ao_navigator->navigate_to_object_mass(
    EXPORTING
      iv_object      = ls_action-semantic_object
      iv_action      = ls_action-semantic_action
      iv_key_field   = ls_action-fieldname
      it_selected_rows = <selected_rows>
    IMPORTING
      ev_refresh     = lv_refresh ).

ELSEIF ls_action-associationname IS NOT INITIAL.
  " Variante 2: Navigation über einen Pfad zu einer Assoziation
  ao_navigator->navigate_via_association(
    iv_cds_view      = av_cds_view
    iv_association_name = ls_action-associationname
    it_parameters     = ao_selection->get_parameters( )
    it_selected_rows  = <selected_rows> ).
```

Aufruf von Aktionen

```
ELSEIF ls_action-data_action IS NOT INITIAL.  
  " Variante 3: Aufruf einer BOPF-Aktion  
  DATA(lv_data_action) = CONV /bobf/obm_name( ls_action-data_action ).  
  
  ao_bopf_handler->execute_action(  
    EXPORTING  
      iv_cds_view      = av_cds_view  
      iv_action        = lv_data_action  
      it_selected_rows = <selected_rows>  
    IMPORTING  
      ev_refresh       = lv_refresh ).  
  
ELSEIF ls_action-url_fieldname IS NOT INITIAL AND <url_field> IS ASSIGNED.  
  " Variante 4: URL im Browser aufrufen  
  me->call_browser( to_lower( <url_field> ) ).  
  
ELSEIF ls_action-send_mail = abap_true.  
  " Variante 5: Feld mit Semantics-Annotation (hier nur E-Mail)  
  me->send_mail_dialog( CONV ad_smtpadr( <field> ) ).
```



Varianten der Intent-Based Navigation

1. Aufruf eines Funktionsbausteins mit Übergabe des Zelleninhalts an einen Importparameter.
2. Erzeugung eines BOR-Objekts mit dem Zelleninhalt als Objektschlüssel und anschließender Aufruf einer Methode des BOR-Objekts.
3. Schreiben des Zelleninhalts in einen Set-/Get-Parameter mit anschließendem Aufruf einer Transaktion

In allen drei Varianten können zusätzliche Parameter durch eine Exit-Klasse gesetzt werden.

Beispiele für Intent-Based Navigation

IntentBased-Navigation im CDS ALV Framework

								
 Semantisches Objekt	Semantische Aktion	Funktionsbaustein	Parametername	Objekttyp	Methode	Transaktionscode	Parameter-Id	
BusinessObject	Display					<u>SWO1</u>	<u>OBJ</u>	
CompanyCode	Display			<u>BUS0002</u>	DISPLAY			
SpoolRequest	Display	<u>COM SE SPOOL DISPLAY</u>	IV_SPOOL_NO					

Event Handling mit Intent-Based Navigation und BOPF-Aktionen

Demonstration (3): Report mit Feldkatalog und Event Handler

[\[DEMO\]](#)

Runtime erzeugen

```
GET TIME STAMP FIELD DATA(lv_timestamp).  
DATA(lv_uuid) = CONV if_sadl_types=>ty_uuid( |ZCDS_ALV:{ iv_cds_view }| ).  
  
DATA(lo_sadl_entity) = cl_sadl_entity_factory=>get_instance( )->get_entity(  
    iv_type = cl_sadl_entity_factory=>co_type-cds  
    iv_id   = CONV sadl_entity_id( iv_cds_view ) ).  
  
DATA(ls_sadl_definition) = cl_sadl_entity_util=>get_sadl_for_entity( lo_sadl_entity ).  
  
DATA(lo_mp) = cl_sadl_mp_factory=>create_mp_entity(  
    iv_uuid          = lv_uuid  
    iv_timestamp     = lv_timestamp  
    is_sadl_definition = ls_sadl_definition ).  
  
DATA(lo_api_factory) = cl_sadl_entity_api_factory=>create( CAST cl_sadl_mp_entity( lo_mp ) ).  
ro_sadl_runtime = lo_api_factory->get_runtime( ).
```

Selektion mit dem SADL-Framework (1)

```
DATA(lo_sadl_runtime) = zcl_salv_cds_ddic_utility=>get_sadl_runtime( av_cds_view ).

DATA(ls_parameters) = VALUE if_sadl_query_engine_types=>ty_parameters(
    entity = VALUE #( (
        entity_alias = cl_sadl_entity_util=>convert( CONV #( av_cds_view ) )
        parameters   = CORRESPONDING #( get_parameters( ) MAPPING name = parname ) ) ) ).

me->get_select_options(
    IMPORTING
        et_field_ranges = DATA(lt_field_ranges)
        ev_maxrec        = DATA(lv_maxrec) ).

LOOP AT lt_field_ranges ASSIGNING FIELD-SYMBOL(<field_range>).
    lo_sadl_runtime->if_sadl_query_fetch~register_condition_provider(
        cl_sadl_cond_prov_factory_pub=>create_basic_condition_factory(
            )->in_range( name      = CONV #( <field_range>-fieldname )
                        t_ranges = CORRESPONDING #( <field_range>-selopt_t ) ) ).
ENDLOOP.
```

Selektion mit dem SADL-Framework (2)

```
DATA(ls_paging) = VALUE if_sadl_query_engine_types=>ty_paging( maximum_rows = lv_maxrec ).

DATA(lt_ddfields) = zcl_salv_cds_ddic_utility=>get_ddic_fields_for_cds_view( av_cds_view ).
DATA(lt_elements) = VALUE if_sadl_query_engine_types=>tt_requested_elements(
    FOR x_ddfield IN lt_ddfields ( CONV #( x_ddfield-fieldname ) ) ).

DATA(ls_requested) = VALUE if_sadl_query_engine_types=>ty_requested(
    elements           = lt_elements
    fill_data          = abap_true
    fill_number_all_hits = abap_true ).

lo_sadl_runtime->fetch(
    EXPORTING
        is_paging           = ls_paging
        is_parameters       = ls_parameters
        is_requested        = ls_requested
    IMPORTING
        et_data_rows       = et_result
        ev_number_all_hits = ev_number_all_hits ).
```

Selektion mit dem SADL-Framework

Demonstration (4): Report mit ALV-Grid, Event Handler und Selektion

[\[DEMO\]](#)

Auswertung von Annotationen für das Selektionsbild

```
CASE <element_annotation>-annoname.  
  WHEN 'CONSUMPTION.FILTER.SELECTIONTYPE'.  
    ls_field_properties-selection_type = <element_annotation>-value.  
  
  WHEN 'CONSUMPTION.FILTER.MULTIPLESELECTIONS'.  
    ls_field_properties-multiple_selections = xsdbool( <element_annotation>-value = 'true' ).  
  
  WHEN 'CONSUMPTION.FILTER.MANDATORY'.  
    ls_field_properties-mandatory = xsdbool( <element_annotation>-value = 'true' ).  
  
  WHEN 'CONSUMPTION.FILTER.DEFAULTVALUE'.  
    ls_field_properties-default_value = <element_annotation>-value.  
  
  WHEN 'CONSUMPTION.VALUEHELP' OR 'CONSUMPTION.VALUEHELPDEFINITION'.  
    ls_field_properties-value_help = abap_true.  
  
  WHEN 'CONSUMPTION.HIDDEN'.  
    ls_field_properties-no_display = xsdbool( <element_annotation>-value = 'true' ).
```

Wirkung einzelner Annotationen (Auszug)

Annotation	Wirkung
Consumption.filter.selectionType	<p data-bbox="542 336 1624 445">Diese Annotation muss angegeben sein, damit ein Feld auf dem Selektionsbild angezeigt wird.</p> <p data-bbox="542 511 1624 620">Der Wert gibt an, welche Vergleichsoperatoren verwendet werden können:</p> <ul data-bbox="542 685 1441 849" style="list-style-type: none">• #SINGLE: nur "EQ" erlaubt• #INTERVAL: nur "EQ" und "BT" erlaubt• #RANGE: alle erlaubt

Wirkung einzelner Annotationen (Auszug)

Annotation	Wirkung
Environment. systemField	Systemfeld als Vorschlagswert (sy-langu, sy-mandt, sy-uname, sy-datum)
Consumption.filter. defaultValue	Festwert als Vorschlagswert für eine Selektion auf dem Selektionsbild
Consumption.filter. mandatory	Selektionsbedingung ist Mussfeld
Consumption.filter. multipleSelections	Gibt an, ob Mehrfachauswahl erlaubt ist
EndUserText.Label	Legt den Selektionstext fest

Generierter Source Code für Datendeklarationen

" 1. Kopfbereich

```
APPEND LINES OF VALUE tt_source(  
  ( |REPORT { iv_progname }.| )  
  ( |TABLES { iv_db_view }. | ) ) TO rt_source.  
append_initial_line.
```

```
APPEND LINES OF VALUE tt_source(  
  ( |CONSTANTS gc_cds_view TYPE ddstrucobjname VALUE '{ iv_cds_view }'.| )  
  ( |CONSTANTS gc_title TYPE sytitle VALUE '{ iv_title }'. | ) ) TO rt_source.  
append_initial_line.
```

```
APPEND LINES OF VALUE tt_source(  
  ( |DATA go_factory TYPE REF TO zif_salv_cds_factory. | )  
  ( |DATA go_controller TYPE REF TO zif_salv_cds_report_controller. | )  
  ( |DATA gt_table TYPE STANDARD TABLE OF { iv_cds_view }. | )  
  ( |DATA gr_table TYPE REF TO data. | )  
  ( |DATA gv_no_end_of_selection TYPE abap_bool. | )  
  ( |DATA gx_exception TYPE REF TO cx_root. | ) ) TO rt_source.  
append_initial_line.
```


Generierter Source Code für die Selektion

```
" 8. Start-Of-Selection (Selektion)
APPEND LINES OF VALUE tt_source(
( | START-OF-SELECTION. | )
( | TRY. | )
( | CLEAR gv_no_end_of_selection. | )
( | go_controller->start_of_selection( | )
( | IMPORTING | )
( | et_output = gt_table | )
( | ev_no_end_of_selection = gv_no_end_of_selection ). | )
( | | )
( | IF gv_no_end_of_selection = abap_true. | )
( | LEAVE LIST-PROCESSING. | )
( | ENDIF. | )
( | CATCH cx_salv_error INTO gx_exception. | )
( | MESSAGE gx_exception TYPE 'I' DISPLAY LIKE 'E'. | )
( | ENDTRY. | )
( | ) ) TO rt_source.
```

Generierter Source Code für die Anzeige

```
" 9. End-Of-Selection (Anzeige)
APPEND LINES OF VALUE tt_source(
( | END-OF-SELECTION. | )
( | TRY. | )
( | go_controller->end_of_selection( CHANGING ct_output = gt_table ). | )
( | CATCH cx_salv_error INTO gx_exception. | )
( | MESSAGE gx_exception TYPE 'I' DISPLAY LIKE 'E'. | )
( | ENDTRY. | ) ) TO rt_source.
```

Generierung des Reports

```
SYNTAX-CHECK FOR lt_source MESSAGE lv_message LINE lv_line WORD lv_word DIRECTORY ENTRY ls_trdir.  
IF sy-subrc <> 0.  
    MESSAGE e006(zisu_salv) WITH iv_cds_view lv_line lv_word INTO av_message.  
    zcx_salv_msg=>raise_exception_from_message( ).  
ENDIF.  
  
INSERT REPORT ev_prognose FROM lt_source.  
UPDATE progdir  
    SET edtx = 'X', occurs = '1'  
    WHERE name = @ev_prognose AND state = 'A'.  
INSERT TEXTPOOL ev_prognose FROM lt_textpool LANGUAGE sy-langu.
```

Demonstration (5): Generierter Report, Parametertransaktion

[\[DEMO\]](#)

Vergleich mit Fiori Elements

Demonstration (6): Fiori Elements App zu demselben View

[\[DEMO\]](#)

- Deutliche Beschleunigung der Reportentwicklung im aktuellen Tagesgeschäft
- Einfacherer Übergang zu S4HANA / Fiori
 - für Entwickler, weil sie sich nur mit einer neuen Technologie auseinandersetzen müssen (erst CDS, dann Fiori)
 - für Anwender, weil sie Anwendungen in der ihnen bekannten GUI erhalten, die aber in ihrer Funktionsweise den künftigen Worklist-Anwendungen in Fiori schon sehr ähnlich sind.
- Weitere Unterstützung von SAP GUI, z.B. für Power User

Vielen Dank für Ihre Aufmerksamkeit!

- Paul Hardy, ABAP to the Future, Rheinwerk Verlag, 3. Auflage 2019
- SAP-Referenz zu CDS-Annotationen: https://help.sap.com/doc/saphelp_nw75/7.5.5/en-US/c2/dd92fb83784c4a87e16e66abeeacbd/frameset.htm

Die Wiedergabe, Vervielfältigung, Verbreitung und/oder Bearbeitung sämtlicher Inhalte und Darstellungen der Publikation sowie jegliche Nutzung – zu welchem Zweck und in welcher Form auch immer – sind nur mit ausdrücklicher Zustimmung durch die rku.it GmbH gestattet.

rku.it oder andere in diesem Dokument erwähnte Produkte oder Dienstleistungen sowie die dazugehörigen Logos sind eingetragene Marken. Es gilt das deutsche Markenrecht.

Die vorliegenden Unterlagen werden von der rku.it GmbH bereitgestellt und dienen ausschließlich zu Informationszwecken. Obwohl mit aller angemessenen Sorgfalt vorgegangen wurde, übernimmt die rku.it GmbH keinerlei Haftung für die Richtigkeit, Aktualität und Vollständigkeit der Inhalte und Darstellungen in dieser Publikation.

Inhalte, auf die die rku.it GmbH verlinkt, werden nicht von der rku.it GmbH bereitgestellt und stellen somit fremde Informationen dar. Eine Haftung für die Richtigkeit, Aktualität und Vollständigkeit dieser verlinkten Informationen wird von der rku.it GmbH nicht übernommen.

In dieser Publikation oder in zugehörigen Präsentationen enthaltene Strategien und etwaige künftige Entwicklungen, Produkte/Dienstleistungen und/oder Plattformen der rku.it GmbH können jederzeit und ohne Angabe von Gründen unangekündigt von der rku.it GmbH geändert werden.

Die in dieser Publikation enthaltenen Informationen stellen keine Garantie oder ein Versprechen dar und begründen keine rechtliche Verpflichtung zur Lieferung von Produkten oder Dienstleistungen. Sämtliche vorrausschauenden Aussagen unterliegen unterschiedlichen Risiken und Unsicherheiten, durch die die tatsächlichen Ergebnisse von den Erwartungen abweichen können. Die vorrausschauenden Aussagen geben die Sicht zu dem Zeitpunkt wieder, zu dem sie getätigt wurden.



Dr. Christopher Graw

rku.it GmbH

Forellstr. 48a, 44269 Herne

Post: Postfach 10 17 09, 44607 Herne

Telefon +49 2323 3688-507

Fax +49 2323 3688-500

Mobil +49 159 04076295

christopher.graw@rku-it.de

www.rku-it.de