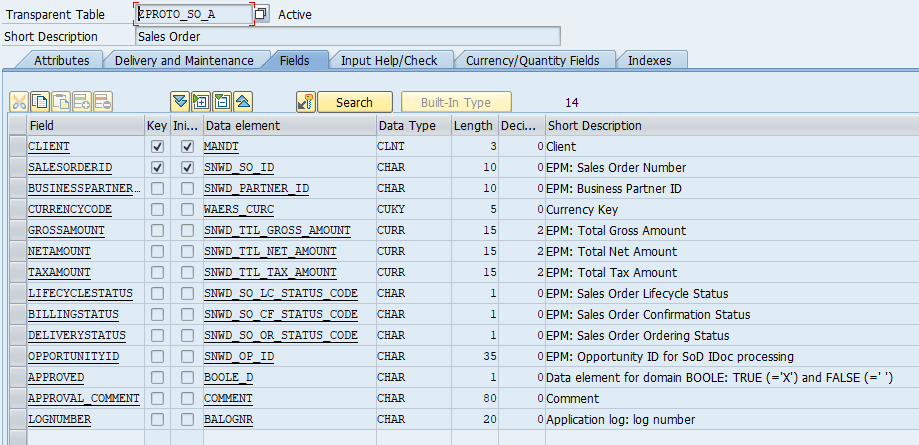
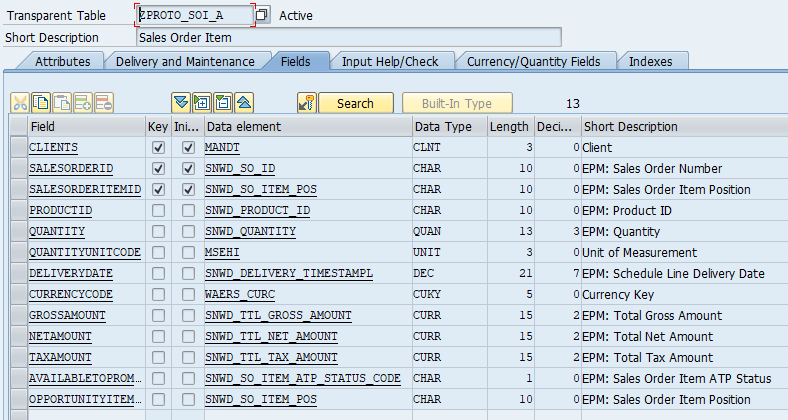
**No** CDS Annotations APP – End to End:

1. Tables and CDS Views (Not necessary to use CDS if not possible):

@AbapCatalog.sqlViewName: 'ZDDL\_I\_ZPROTOSO'

@AbapCatalog.compiler.compareFilter: true

@AbapCatalog.preserveKey: true

@AccessControl.authorizationCheck: #NOT\_REQUIRED

@Search.searchable: true

@EndUserText.label: 'CDS View for ZPROTO\_SO\_A BASIC'

**define** **view** ZDDL\_I\_ZPROTO\_SO\_A

**as** **select** **from** zproto\_so\_a **as** SalesOrder

**association** **[**1**..\*]** **to** ZDDL\_I\_ZPROTO\_SOI\_A **as** \_Item

**on** **$projection.**SalesOrderID **=** \_Item**.**SalesOrderID

**{**

//SalesOrder

@Search.defaultSearchElement: true

**key** salesorderid **as** SalesOrderID**,**

@Search.defaultSearchElement: true

businesspartnerid **as** BusinessPartnerID**,**

currencycode **as** CurrencyCode**,**

grossamount **as** GrossAmount**,**

netamount **as** NetAmount**,**

taxamount **as** TaxAmount**,**

lifecyclestatus **as** LifeCycleStatus**,**

// Associations/Navigation

\_Item

**}**

@AbapCatalog.sqlViewName: 'ZDDL\_I\_ZPROTOSOI'

@AbapCatalog.compiler.compareFilter: true

@AccessControl.authorizationCheck: #NOT\_REQUIRED

@EndUserText.label: 'CDS View for ZPROTO\_SO\_A'

@Search.searchable: true

**define** **view** ZDDL\_I\_ZPROTO\_SOI\_A

**as** **select** **from** zproto\_soi\_a **as** Item

//Association to header

**association** **[**1**..**1**]** **to** ZDDL\_I\_ZPROTO\_SO\_A **as** \_SalesOrder

**on** **$projection.**SalesOrderID **=** \_SalesOrder**.**SalesOrderID

**{**

//Item

@Search.defaultSearchElement: true

@ObjectModel.readOnly: true

**key** salesorderid **as** SalesOrderID**,**

@Search.defaultSearchElement: true

**key** salesorderitemid **as** SalesOrderItemID**,**

@Search.defaultSearchElement: true

productid **as** ProductID**,**

quantity **as** Quantity**,**

quantityunitcode **as** QuantityUnitCode**,**

deliverydate **as** DeliveryDate**,**

currencycode **as** CurrencyCode**,**

grossamount **as** GrossAmount**,**

netamount **as** NetAmount**,**

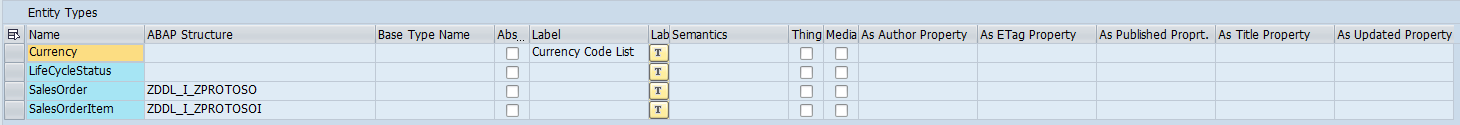
taxamount **as** TaxAmount**,**

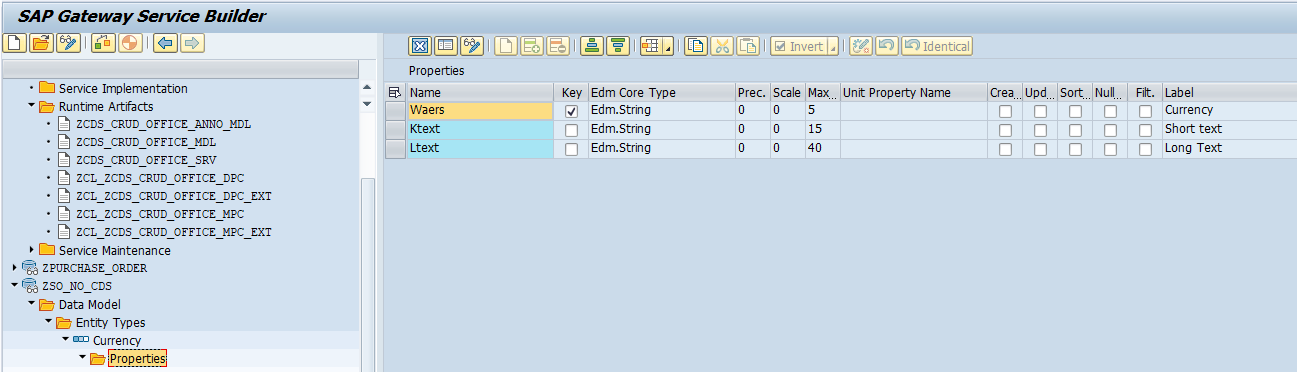
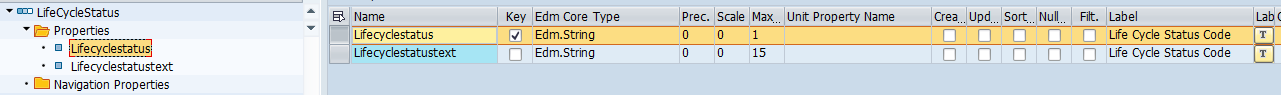
//Association to SO Order

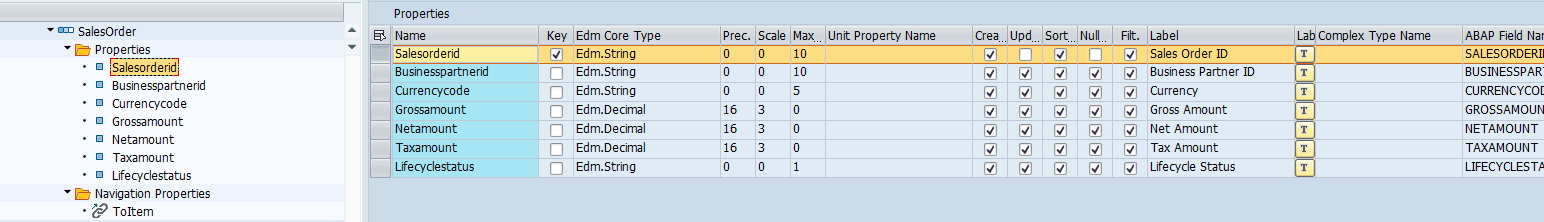
\_SalesOrder

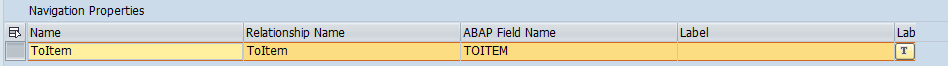
**}**

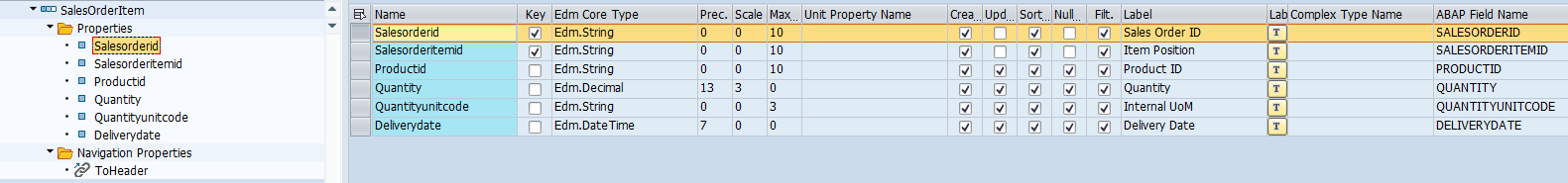
1. OData Service
   1. Entity Types

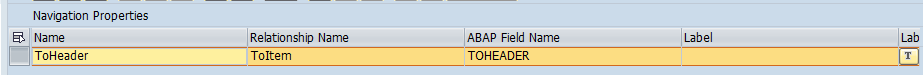


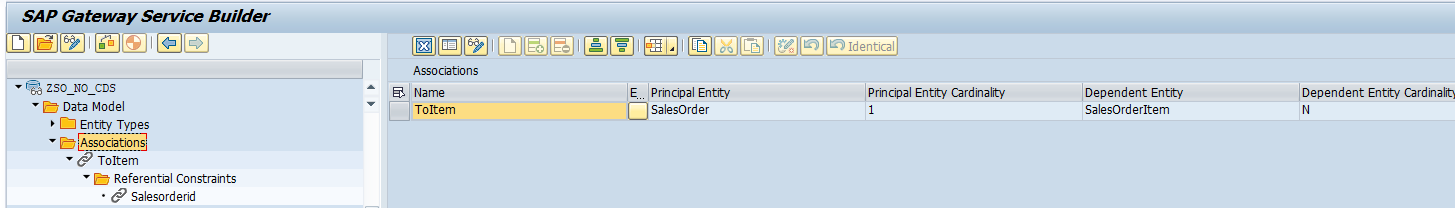
  
  


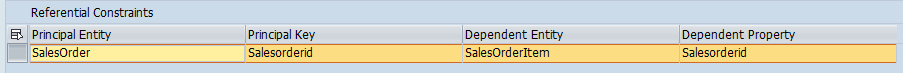


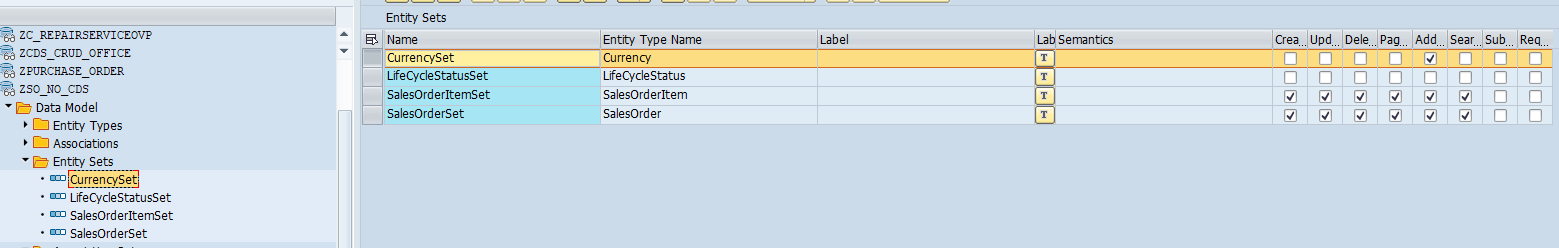
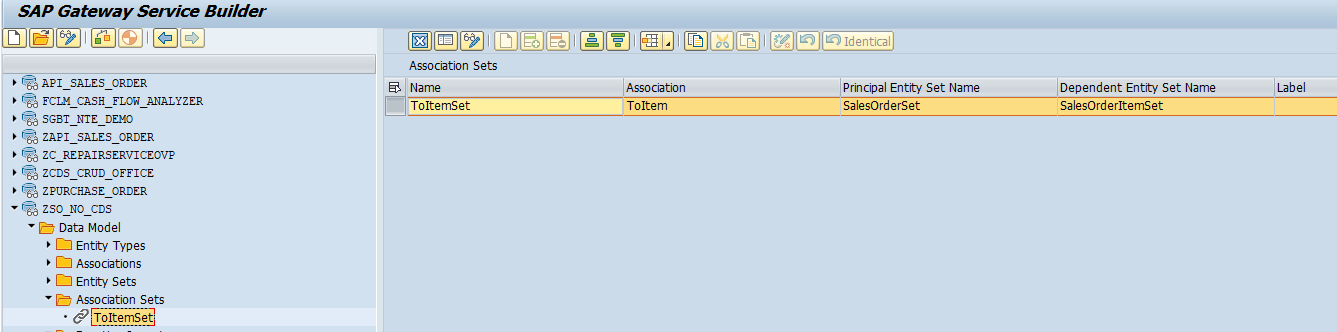


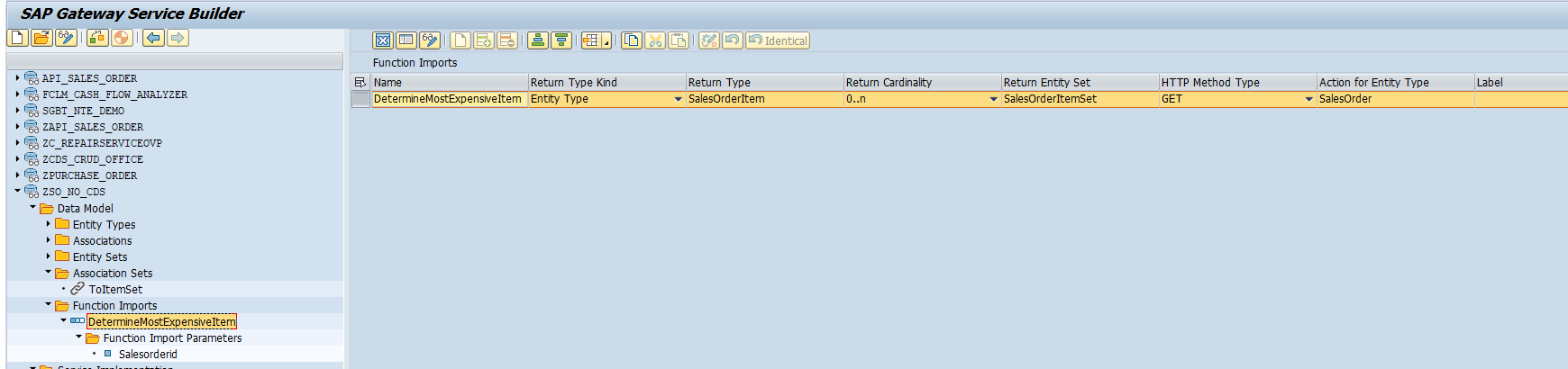


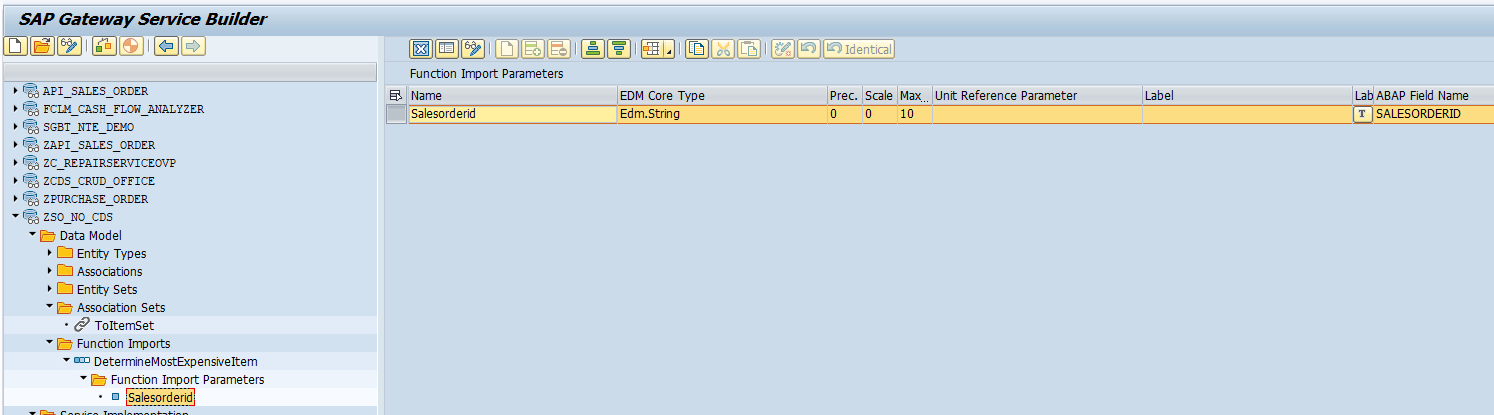


* 1. Associations  
       
     

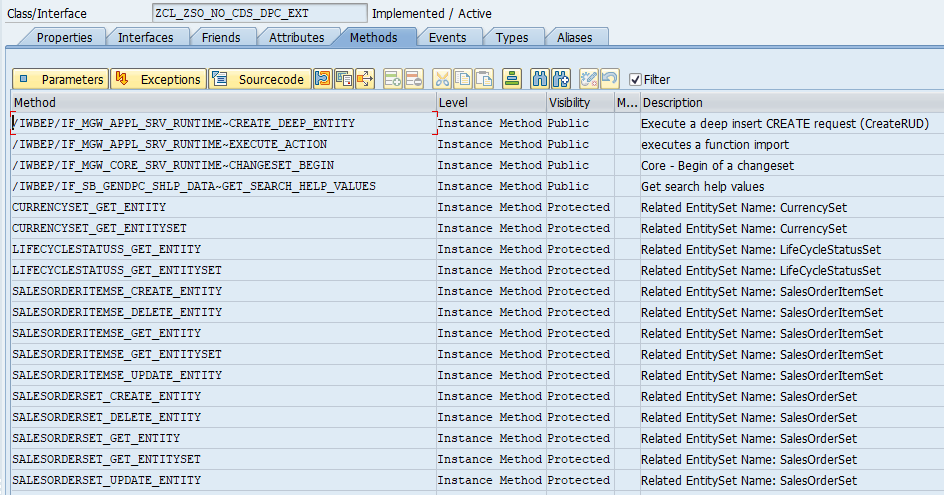


* 1. Entity Set
  2. Associations Set  
       
     
  3. Function Imports





* 1. Run Artifacts ( DPC\_EXT Class )



METHOD /iwbep/if\_mgw\_appl\_srv\_runtime~create\_deep\_entity.  
  
  DATA: BEGIN OF ls\_order\_item\_data.  
      INCLUDE TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorder.  
  DATA: toitem TYPE zcl\_zso\_no\_cds\_mpc=>tt\_salesorderitem,  
        END OF ls\_order\_item\_data.  
  
  
  DATA: ls\_so\_db  TYPE zproto\_so\_a,  
        lt\_soi\_db TYPE TABLE OF zproto\_soi\_a,  
        ls\_soi\_db TYPE zproto\_soi\_a.  
  
  DATA(lv\_entityset) = io\_tech\_request\_context->get\_entity\_set\_name( ).  
  
  CASE lv\_entityset.  
    WHEN 'SalesOrderSet'.  
  
      io\_data\_provider->read\_entry\_data(  
        IMPORTING  
          es\_data = ls\_order\_item\_data ).  
  
      IF ls\_order\_item\_data-salesorderid IS INITIAL.  
        "Ideale wäre mit SNRO transaktion machen  
        SELECT COUNT(\*)  
          INTO @DATA(lv\_count)  
          FROM zddl\_i\_zproto\_so\_a.  
        IF sy-subrc = 0.  
          ls\_order\_item\_data-salesorderid = lv\_count + 1.  
        ELSE.  
          ls\_order\_item\_data-salesorderid = 1.  
        ENDIF.  
        ls\_order\_item\_data-salesorderid = |{ ls\_order\_item\_data-salesorderid ALPHA = IN }|.  
      ENDIF.  
  
      ls\_so\_db  = CORRESPONDING #( ls\_order\_item\_data ).  
  
      INSERT zproto\_so\_a FROM ls\_so\_db.  
      IF sy-subrc <> 0.  
        ROLLBACK WORK.  
  
        DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
        "Duplicated KEY - Not allowed  
        lo\_message\_container->add\_message(  
          EXPORTING  
            iv\_msg\_type               = 'E'  
            iv\_msg\_id                 = 'ZIMU'  
            iv\_msg\_number             =  004  ).  
  
        RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
          EXPORTING  
            message\_container = lo\_message\_container.  
  
      ENDIF.  
      " Insert ITEMS  
      LOOP AT ls\_order\_item\_data-toitem ASSIGNING FIELD-SYMBOL(<fs\_item>).  
        DATA(lv\_tabix) = sy-tabix.  
        <fs\_item>-salesorderid = ls\_order\_item\_data-salesorderid.  
  
        IF <fs\_item>-salesorderitemid IS INITIAL.  
          <fs\_item>-salesorderitemid = lv\_tabix \* 1000.  
          <fs\_item>-salesorderitemid = |{ <fs\_item>-salesorderitemid ALPHA = IN }|.  
        ENDIF.  
  
        ls\_soi\_db = CORRESPONDING #( <fs\_item> ).  
        INSERT ls\_soi\_db INTO TABLE lt\_soi\_db.  
        CLEAR ls\_soi\_db.  
      ENDLOOP.  
  
      INSERT zproto\_soi\_a FROM TABLE lt\_soi\_db.  
      IF sy-subrc <> 0.  
        ROLLBACK WORK.  
  
        lo\_message\_container = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
        "Duplicated KEY - Not allowed  
        lo\_message\_container->add\_message(  
          EXPORTING  
            iv\_msg\_type               = 'E'  
            iv\_msg\_id                 = 'ZIMU'  
            iv\_msg\_number             =  004  ).  
  
        RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
          EXPORTING  
            message\_container = lo\_message\_container.  
  
      ENDIF.  
  
      copy\_data\_to\_ref(  
        EXPORTING  
          is\_data = ls\_order\_item\_data  
        CHANGING  
          cr\_data = er\_deep\_entity  ).  
  
    WHEN OTHERS.  
  ENDCASE.  
  
ENDMETHOD.

METHOD /iwbep/if\_mgw\_appl\_srv\_runtime~execute\_action.  
  
  DATA: ls\_so  TYPE zcl\_zso\_no\_cds\_mpc\_ext=>ts\_salesorder,  
        lt\_soi TYPE TABLE OF zcl\_zso\_no\_cds\_mpc\_ext=>ts\_salesorderitem,  
        ls\_soi TYPE zcl\_zso\_no\_cds\_mpc\_ext=>ts\_salesorderitem.  
  
  DATA(lv\_action) = io\_tech\_request\_context->get\_function\_import\_name( ).  
  
  CASE lv\_action.  
    WHEN 'DetermineMostExpensiveItem'.  
  
      io\_tech\_request\_context->get\_converted\_parameters(  
        IMPORTING  
          es\_parameter\_values = ls\_so  ).  
  
      ls\_so-salesorderid = |{ ls\_so-salesorderid ALPHA = IN }|.  
  
      SELECT \*  
        FROM zddl\_i\_zproto\_soi\_a  
        INTO TABLE @DATA(lt\_zddl\_i\_zproto\_soi\_a)  
        WHERE salesorderid = @ls\_so-salesorderid ORDER BY grossamount DESCENDING.  
      IF sy-subrc = 0.  
        READ TABLE lt\_zddl\_i\_zproto\_soi\_a INTO DATA(ls\_item) INDEX 1.  
  
        ls\_soi = CORRESPONDING #( ls\_item ).  
  
        "Format TimeStamp  
        DATA: lv\_tstm TYPE cest1-timestmp,  
              lv\_date TYPE sy-datum,  
              lv\_time TYPE sy-uzeit.  
  
        IF ls\_soi-deliverydate IS NOT INITIAL.  
  
          lv\_tstm = ls\_soi-deliverydate.  
  
          CALL FUNCTION 'RKE\_TIMESTAMP\_CONVERT\_OUTPUT'  
            EXPORTING  
              i\_dayst    = sy-dayst  
              i\_timestmp = lv\_tstm  
              i\_tzone    = sy-tzone  
            IMPORTING  
              e\_date     = lv\_date  
              e\_time     = lv\_time.  
  
          ls\_soi-deliverydate = lv\_date && lv\_time.  
  
        ENDIF.  
  
        INSERT ls\_soi INTO TABLE lt\_soi.  
  
        copy\_data\_to\_ref(  
          EXPORTING  
            is\_data = lt\_soi  
          CHANGING  
            cr\_data = er\_data ).  
  
      ELSE.  
        RAISE EXCEPTION TYPE /IWBEP/CX\_MGW\_BUSI\_EXCEPTION.  
      ENDIF.  
  
    WHEN OTHERS.  
  ENDCASE.  
  
ENDMETHOD.

METHOD /iwbep/if\_mgw\_core\_srv\_runtime~changeset\_begin.  
  
  TRY.  
      CALL METHOD super->/iwbep/if\_mgw\_core\_srv\_runtime~changeset\_begin  
        EXPORTING  
          it\_operation\_info  = it\_operation\_info  
          it\_changeset\_input = it\_changeset\_input  
        CHANGING  
          cv\_defer\_mode      = cv\_defer\_mode.  
    CATCH /iwbep/cx\_mgw\_busi\_exception /iwbep/cx\_mgw\_tech\_exception.  
  
      "Updating more than 2 entries ?  
  
  ENDTRY.  
  
  
  
ENDMETHOD.

method CURRENCYSET\_GET\_ENTITY.  
  
  DATA: lt\_curr  TYPE zcl\_zso\_no\_cds\_mpc=>tt\_currency,  
        ls\_curr  TYPE LINE OF zcl\_zso\_no\_cds\_mpc=>tt\_currency,  
        lr\_curr  TYPE RANGE OF waers\_curc,  
        ls\_cur   LIKE LINE OF lr\_curr.  
  
  LOOP AT it\_key\_tab INTO DATA(ls\_key\_tab).  
  
    CASE ls\_key\_tab-name.  
      WHEN 'Currency'.  
        ls\_cur-sign   = 'I'.  
        ls\_cur-option = 'EQ'.  
        ls\_cur-low    = ls\_key\_tab-value.  
        CONDENSE ls\_cur-low.  
        INSERT ls\_cur INTO TABLE lr\_curr.  
      WHEN OTHERS.  
    ENDCASE.  
  
  ENDLOOP.  
  
  SELECT SINGLE a~waers, b~ktext , b~ltext  
    INTO @ls\_curr  
    FROM tcurc AS a  
    INNER JOIN tcurt AS b ON ( a~waers = b~waers )  
    WHERE a~waers IN @lr\_curr  
      AND b~spras EQ @sy-langu.  
  IF sy-subrc = 0.  
    er\_entity = ls\_curr.  
  ENDIF.  
  
endmethod.

METHOD currencyset\_get\_entityset.  
  
  DATA: lt\_curr TYPE zcl\_zso\_no\_cds\_mpc=>tt\_currency,  
        ls\_curr TYPE LINE OF zcl\_zso\_no\_cds\_mpc=>tt\_currency,  
        lr\_curr TYPE RANGE OF waers\_curc,  
        ls\_cur  LIKE LINE OF lr\_curr,  
        lt\_sh   TYPE ddshselops,  
        ls\_sh   TYPE LINE OF ddshselops.  
  
  
  DATA(lr\_filter) = io\_tech\_request\_context->get\_filter( ).  
  IF lr\_filter IS BOUND.  
    DATA(lt\_filter\_so) = lr\_filter->get\_filter\_select\_options( ).  
  ENDIF.  
  
  IF lt\_filter\_so IS NOT INITIAL.  
  
    LOOP AT lt\_filter\_so ASSIGNING FIELD-SYMBOL(<fs\_filter\_so>).  
      ls\_sh-shlpname  = 'FC\_WAERS'.  
      ls\_sh-shlpfield = <fs\_filter\_so>-property.  
      LOOP AT <fs\_filter\_so>-select\_options ASSIGNING FIELD-SYMBOL(<fs\_so>).  
        ls\_sh-sign   = <fs\_so>-sign.  
        ls\_sh-low    = <fs\_so>-low.  
        ls\_sh-high   = <fs\_so>-high.  
        ls\_sh-option = <fs\_so>-option.  
        INSERT ls\_sh INTO TABLE lt\_sh.  
        CLEAR: ls\_sh-sign, ls\_sh-low, ls\_sh-high, ls\_sh-option.  
      ENDLOOP.  
    ENDLOOP.  
  
    me->/iwbep/if\_sb\_gendpc\_shlp\_data~get\_search\_help\_values( EXPORTING iv\_shlp\_name   = 'FC\_WAERS'  
                                                                        it\_selopt      = lt\_sh  
                                                              IMPORTING et\_return\_list = DATA(lt\_filter) ).  
  
    LOOP AT lt\_filter ASSIGNING FIELD-SYMBOL(<fs\_filter>).  
      CASE <fs\_filter>-field\_name.  
        WHEN 'WAERS'.  
          ls\_curr-waers = <fs\_filter>-field\_value.  
        WHEN 'KTEXT'.  
          ls\_curr-ktext = <fs\_filter>-field\_value.  
        WHEN 'LTEXT'.  
          ls\_curr-ltext = <fs\_filter>-field\_value.  
        WHEN OTHERS.  
      ENDCASE.  
  
      AT END OF record\_number.  
        INSERT ls\_curr INTO TABLE lt\_curr.  
      ENDAT.  
    ENDLOOP.  
  
    et\_entityset = lt\_curr.  
  
  ELSE.  
  
    LOOP AT it\_key\_tab INTO DATA(ls\_key\_tab).  
  
      CASE ls\_key\_tab-name.  
        WHEN 'Currency'.  
          ls\_cur-sign   = 'I'.  
          ls\_cur-option = 'EQ'.  
          ls\_cur-low    = ls\_key\_tab-value.  
          CONDENSE ls\_cur-low.  
          INSERT ls\_cur INTO TABLE lr\_curr.  
        WHEN OTHERS.  
      ENDCASE.  
  
    ENDLOOP.  
  
    SELECT a~waers, b~ktext , b~ltext  
      INTO TABLE @lt\_curr  
      FROM tcurc AS a  
      INNER JOIN tcurt AS b ON ( a~waers = b~waers )  
      WHERE a~waers IN @lr\_curr  
        AND b~spras EQ @sy-langu.  
    IF sy-subrc = 0.  
      et\_entityset = lt\_curr.  
    ENDIF.  
  
  ENDIF.  
  
ENDMETHOD.

METHOD lifecyclestatuss\_get\_entity.  
  
  DATA: lv\_so    TYPE zddl\_i\_zproto\_so\_a-salesorderid,  
        lt\_dd07v TYPE TABLE OF dd07v.  
  
  READ TABLE it\_key\_tab INTO DATA(ls\_key) WITH KEY name = 'Salesorderid'.  
  IF sy-subrc = 0.  
    lv\_so = |{ ls\_key-value ALPHA = IN }|.  
  ENDIF.  
  
  SELECT SINGLE \*  
    FROM zddl\_i\_zproto\_so\_a  
    INTO @DATA(ls\_so\_db)  
    WHERE salesorderid = @lv\_so.  
  
  IF sy-subrc = 0.  
  
    CALL FUNCTION 'DD\_DOMVALUES\_GET'  
      EXPORTING  
        domname        = 'D\_SO\_LC'  
        text           = 'X'  
        langu          = sy-langu  
      TABLES  
        dd07v\_tab      = lt\_dd07v  
      EXCEPTIONS  
        wrong\_textflag = 1  
        OTHERS         = 2.  
  
    READ TABLE lt\_dd07v INTO DATA(ls\_dd07v) WITH KEY domvalue\_l = ls\_so\_db-lifecyclestatus.  
    IF sy-subrc = 0.  
      er\_entity-lifecyclestatus     = ls\_dd07v-domvalue\_l.  
      er\_entity-lifecyclestatustext = ls\_dd07v-ddtext.  
    ENDIF.  
  
  ENDIF.  
  
ENDMETHOD.

METHOD lifecyclestatuss\_get\_entityset.  
  
  DATA: lt\_life  TYPE zcl\_zso\_no\_cds\_mpc=>tt\_lifecyclestatus,  
        lt\_dd07v TYPE TABLE OF dd07v,  
        ls\_life  TYPE LINE OF zcl\_zso\_no\_cds\_mpc=>tt\_lifecyclestatus.  
  
  CALL FUNCTION 'DD\_DOMVALUES\_GET'  
    EXPORTING  
      domname        = 'D\_SO\_LC'  
      text           = 'X'  
      langu          = sy-langu  
    TABLES  
      dd07v\_tab      = lt\_dd07v  
    EXCEPTIONS  
      wrong\_textflag = 1  
      OTHERS         = 2.  
  
  LOOP AT lt\_dd07v ASSIGNING FIELD-SYMBOL(<fs\_dd07v>).  
    ls\_life-lifecyclestatus     = <fs\_dd07v>-domvalue\_l.  
    ls\_life-lifecyclestatustext = <fs\_dd07v>-ddtext.  
    INSERT ls\_life INTO TABLE lt\_life.  
  ENDLOOP.  
  
  ET\_ENTITYSET = lt\_life.  
  
ENDMETHOD.

METHOD salesorderitemse\_create\_entity.  
  
  DATA: ls\_so    TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorderitem,  
        ls\_so\_db TYPE zproto\_soi\_a.  
  
  io\_data\_provider->read\_entry\_data(  
    IMPORTING  
      es\_data = ls\_so ).  
  
  "It is creating a new Item on the UI  
  READ TABLE it\_key\_tab INTO DATA(ls\_key) WITH KEY name = 'Salesorderid'.  
  IF sy-subrc = 0 AND ls\_so-salesorderid IS INITIAL.  
    ls\_so-salesorderid = ls\_key-value.  
    ls\_so-salesorderid = |{ ls\_so-salesorderid  ALPHA = IN }|.  
  ENDIF.  
  
  SELECT SINGLE salesorderid  
         FROM zddl\_i\_zproto\_so\_a  
         INTO @DATA(lv\_so\_id)  
         WHERE salesorderid = @ls\_so-salesorderid.  
  IF sy-subrc <> 0.  
    mo\_context->get\_message\_container( )->add\_message\_text\_only(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_text               = 'Create Header First' ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception.  
  
  ELSE.  
    "Calculate next item  
    SELECT salesorderid, salesorderitemid  
      FROM zddl\_i\_zproto\_soi\_a  
      INTO TABLE @DATA(lt\_items)  
      WHERE salesorderid = @ls\_so-salesorderid ORDER BY salesorderitemid DESCENDING.  
    IF sy-subrc = 0.  
      READ TABLE lt\_items ASSIGNING FIELD-SYMBOL(<fs\_item>) INDEX 1.  
      IF sy-subrc = 0.  
        ls\_so-salesorderitemid = <fs\_item>-salesorderitemid + 1000.  
        ls\_so-salesorderitemid = |{ ls\_so-salesorderitemid  ALPHA = IN }|.  
      ENDIF.  
    ELSE. " First Item  
      ls\_so-salesorderitemid = 1000.  
      ls\_so-salesorderitemid = |{ ls\_so-salesorderitemid  ALPHA = IN }|.  
    ENDIF.  
  ENDIF.  
  
  ls\_so\_db  = CORRESPONDING #( ls\_so ).  
  
  INSERT zproto\_soi\_a FROM ls\_so\_db.  
  IF sy-subrc <> 0.  
  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    "Duplicated Key  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  004  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  
  ENDIF.  
  
  er\_entity = CORRESPONDING #( ls\_so ).  
  
ENDMETHOD.

METHOD salesorderitemse\_delete\_entity.  
  
  DATA: ls\_so TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorderitem.  
  
  io\_tech\_request\_context->get\_converted\_keys(  
    IMPORTING  
      es\_key\_values = ls\_so ).  
  
  SELECT COUNT(\*)  
     FROM zproto\_soi\_a  
     INTO @DATA(lv\_items)  
     WHERE salesorderid     = @ls\_so-salesorderid.  
  
  " Update the table rather than the CDS view itself.  
  DELETE FROM zproto\_soi\_a WHERE salesorderid     = ls\_so-salesorderid  
                             AND salesorderitemid = ls\_so-salesorderitemid.  
  IF sy-subrc <> 0.  
  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    "Data does not exist.  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  005  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  
  ENDIF.  
  
  
ENDMETHOD.

METHOD salesorderitemse\_get\_entity.  
  
  DATA: ls\_soi TYPE zddl\_i\_zproto\_soi\_a.  
  
  io\_tech\_request\_context->get\_converted\_keys(  
    IMPORTING  
      es\_key\_values = ls\_soi " Entity Key Values - converted  
  ).  
  
  SELECT SINGLE salesorderid, salesorderitemid, productid, quantity, quantityunitcode, deliverydate  
    FROM zddl\_i\_zproto\_soi\_a  
    INTO @DATA(ls\_soi\_db)  
    WHERE salesorderid EQ @ls\_soi-salesorderid  
      AND salesorderitemid EQ @ls\_soi-salesorderitemid.  
  
  IF sy-subrc = 0.  
  
    er\_entity = CORRESPONDING #( ls\_soi\_db ).  
  
    "Format TimeStamp  
    DATA: lv\_tstm TYPE cest1-timestmp,  
          lv\_date TYPE sy-datum,  
          lv\_time TYPE sy-uzeit.  
  
    IF er\_entity-deliverydate IS NOT INITIAL.  
  
      lv\_tstm = er\_entity-deliverydate.  
  
      CALL FUNCTION 'RKE\_TIMESTAMP\_CONVERT\_OUTPUT'  
        EXPORTING  
          i\_dayst    = sy-dayst  
          i\_timestmp = lv\_tstm  
          i\_tzone    = sy-tzone  
        IMPORTING  
          e\_date     = lv\_date  
          e\_time     = lv\_time.  
  
      er\_entity-deliverydate = lv\_date && lv\_time.  
  
    ENDIF.  
  
  ELSE.  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    "No Data Found  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  003  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  ENDIF.  
ENDMETHOD.

METHOD salesorderitemse\_get\_entityset.  
  
  TYPES: ty\_salesorderid     TYPE RANGE OF zddl\_i\_zproto\_soi\_a-salesorderid,  
         ty\_salesorderitemid TYPE RANGE OF zddl\_i\_zproto\_soi\_a-salesorderitemid,  
         ty\_productid        TYPE RANGE OF zddl\_i\_zproto\_soi\_a-productid,  
         ty\_quantity         TYPE RANGE OF zddl\_i\_zproto\_soi\_a-quantity,  
         ty\_quantityunitcode TYPE RANGE OF zddl\_i\_zproto\_soi\_a-quantityunitcode,  
         ty\_deliverydate     TYPE RANGE OF zddl\_i\_zproto\_soi\_a-deliverydate.  
  
  DATA: lr\_salesorderid     TYPE ty\_salesorderid,  
        lr\_salesorderitemid TYPE ty\_salesorderitemid,  
        lr\_productid        TYPE ty\_productid,  
        lr\_quantity         TYPE ty\_quantity,  
        lr\_quantityunitcode TYPE ty\_quantityunitcode,  
        lr\_deliverydate     TYPE ty\_deliverydate.  
  
  DATA: lt\_sort TYPE abap\_sortorder\_tab,  
        ls\_sort TYPE LINE OF abap\_sortorder\_tab.  
  
  DATA(lr\_filter) = io\_tech\_request\_context->get\_filter( ).  
  IF lr\_filter IS BOUND.  
    DATA(lt\_filter\_so) = lr\_filter->get\_filter\_select\_options( ).  
  ENDIF.  
  
  "URI Filter <> $filter  
  LOOP AT it\_key\_tab INTO DATA(ls\_key\_tab).  
  
    TRANSLATE ls\_key\_tab-name TO UPPER CASE.  
  
    CASE ls\_key\_tab-name.  
      WHEN 'SALESORDERID'.  
        lr\_salesorderid[]      = VALUE ty\_salesorderid( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
        LOOP AT lr\_salesorderid ASSIGNING FIELD-SYMBOL(<fs\_so>).  
          <fs\_so>-low  =  |{ <fs\_so>-low  ALPHA = IN }|.  
          <fs\_so>-high =  |{ <fs\_so>-high ALPHA = IN }|.  
        ENDLOOP.  
      WHEN 'SALESORDERITEMID'.  
        lr\_salesorderitemid[] = VALUE ty\_salesorderitemid( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'PRODUCTID'.  
        lr\_productid[]        = VALUE ty\_productid( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'QUANTITY'.  
        lr\_quantity[]         = VALUE ty\_quantity( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'QUANTITYUNITCODE'.  
        lr\_quantityunitcode[] = VALUE ty\_quantityunitcode( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'DELIVERYDATE'.  
        lr\_deliverydate[]     = VALUE ty\_deliverydate( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN OTHERS.  
    ENDCASE.  
  
  ENDLOOP.  
  
  "This is useful for dynamic queries. $filter  
  DATA(lv\_sql) = io\_tech\_request\_context->get\_osql\_where\_clause\_convert( ).  
  DATA(lv\_sql2) = io\_tech\_request\_context->get\_osql\_where\_clause( ).  
  
  " $filter  
  LOOP AT lt\_filter\_so REFERENCE INTO DATA(lr\_filter\_so).  
    CASE lr\_filter\_so->property.  
      WHEN 'SALESORDERID'.  
        lr\_salesorderid[]      = CORRESPONDING #( lr\_filter\_so->select\_options ).  
        LOOP AT lr\_salesorderid ASSIGNING <fs\_so>.  
          <fs\_so>-low  =  |{ <fs\_so>-low  ALPHA = IN }|.  
          <fs\_so>-high =  |{ <fs\_so>-high ALPHA = IN }|.  
        ENDLOOP.  
      WHEN 'SALESORDERITEMID'.  
        lr\_salesorderitemid[] = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'PRODUCTID'.  
        lr\_productid[]        = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'QUANTITY'.  
        lr\_quantity[]         = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'QUANTITYUNITCODE'.  
        lr\_quantityunitcode[] = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'DELIVERYDATE'.  
        lr\_deliverydate[]     = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN OTHERS.  
    ENDCASE.  
  ENDLOOP.  
  
  SELECT salesorderid, salesorderitemid, productid, quantity, quantityunitcode, deliverydate  
    FROM zddl\_i\_zproto\_soi\_a  
    INTO TABLE @DATA(lt\_so)  
    WHERE salesorderid     IN @lr\_salesorderid  
      AND salesorderitemid IN @lr\_salesorderitemid  
      AND productid        IN @lr\_productid  
      AND quantity         IN @lr\_quantity  
      AND quantityunitcode IN @lr\_quantityunitcode  
      AND deliverydate     IN @lr\_deliverydate.  
  
  IF sy-subrc = 0.  
  
    " $orderby  
    DATA(lt\_orderby) = io\_tech\_request\_context->get\_orderby( ).  
    LOOP AT lt\_orderby INTO DATA(ls\_orderby).  
      ls\_sort-name = ls\_orderby-property.  
      IF ls\_orderby-order = 'desc'.  
        ls\_sort-descending = abap\_true.  
      ENDIF.  
      INSERT ls\_sort INTO TABLE lt\_sort.  
      CLEAR ls\_sort.  
    ENDLOOP.  
  
    SORT lt\_so BY (lt\_sort).  
  
    " $inlinecount  
    IF io\_tech\_request\_context->has\_inlinecount( ).  
      es\_response\_context-inlinecount = lines( lt\_so ).  
    ENDIF.  
    " $skip  
    IF is\_paging-skip IS NOT INITIAL.  
      DO is\_paging-skip TIMES.  
        DELETE lt\_so[] INDEX 1.  
      ENDDO.  
    ENDIF.  
    " $top  
    IF is\_paging-top IS NOT INITIAL.  
      LOOP AT lt\_so INTO DATA(ls\_so).  
        IF sy-tabix > is\_paging-top.  
          DELETE lt\_so INDEX sy-tabix.  
        ENDIF.  
      ENDLOOP.  
    ENDIF.  
  
    et\_entityset[] = CORRESPONDING #( lt\_so ).  
  
    "Format TimeStamp  
    DATA: lv\_tstm TYPE cest1-timestmp,  
          lv\_date TYPE sy-datum,  
          lv\_time TYPE sy-uzeit.  
  
    LOOP AT et\_entityset REFERENCE INTO DATA(lo\_set).  
      IF lo\_set->deliverydate IS NOT INITIAL.  
  
        lv\_tstm = lo\_set->deliverydate.  
  
        CALL FUNCTION 'RKE\_TIMESTAMP\_CONVERT\_OUTPUT'  
          EXPORTING  
            i\_dayst    = sy-dayst  
            i\_timestmp = lv\_tstm  
            i\_tzone    = sy-tzone  
          IMPORTING  
            e\_date     = lv\_date  
            e\_time     = lv\_time.  
  
        lo\_set->deliverydate = lv\_date && lv\_time.  
  
      ENDIF.  
    ENDLOOP.  
  
  ELSE.  
    "It is not a Issue to not have ITEMS  
\*    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception.  
  ENDIF.  
  
ENDMETHOD.

METHOD salesorderitemse\_update\_entity.  
  
  DATA: ls\_so    TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorderitem,  
        ls\_so\_db TYPE zproto\_soi\_a.  
  
  io\_data\_provider->read\_entry\_data(  
    IMPORTING  
      es\_data = ls\_so ).  
  
  ls\_so\_db  = CORRESPONDING #( ls\_so ).  
  er\_entity = CORRESPONDING #( ls\_so ).  
  
  UPDATE zproto\_soi\_a FROM ls\_so\_db.  
  IF sy-subrc <> 0.  
  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    "Error during update  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  006  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  
  ENDIF.  
  
ENDMETHOD.

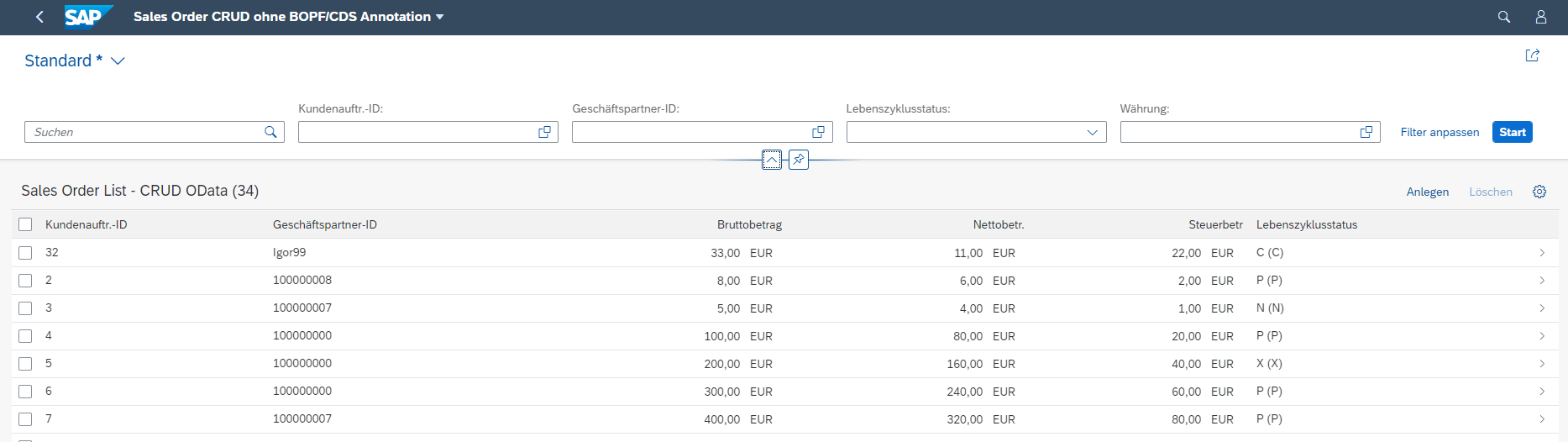
METHOD salesorderset\_create\_entity.  
  
  DATA: ls\_so    TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorder,  
        ls\_so\_db TYPE zproto\_so\_a.  
  
  io\_data\_provider->read\_entry\_data(  
    IMPORTING  
      es\_data = ls\_so ).  
  
  IF ls\_so-salesorderid IS INITIAL.  
    "Ideale wäre mit SNRO transaktion machen  
    SELECT COUNT(\*)  
      INTO @DATA(lv\_count)  
      FROM zddl\_i\_zproto\_so\_a.  
    IF sy-subrc = 0.  
      ls\_so-salesorderid = lv\_count + 1.  
    ELSE.  
      ls\_so-salesorderid = 1.  
    ENDIF.  
    ls\_so-salesorderid = |{ ls\_so-salesorderid ALPHA = IN }|.  
  ELSE.  
    ls\_so-salesorderid = |{ ls\_so-salesorderid ALPHA = IN }|.  
  ENDIF.  
  
  ls\_so\_db  = CORRESPONDING #( ls\_so ).  
  
  INSERT zproto\_so\_a FROM ls\_so\_db.  
  IF sy-subrc <> 0.  
  
    mo\_context->get\_message\_container( )->add\_message\_text\_only(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_text               = 'Duplicated KEY' ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception.  
  
  ELSE.  
    er\_entity = CORRESPONDING #( ls\_so ).  
  ENDIF.  
  
ENDMETHOD.

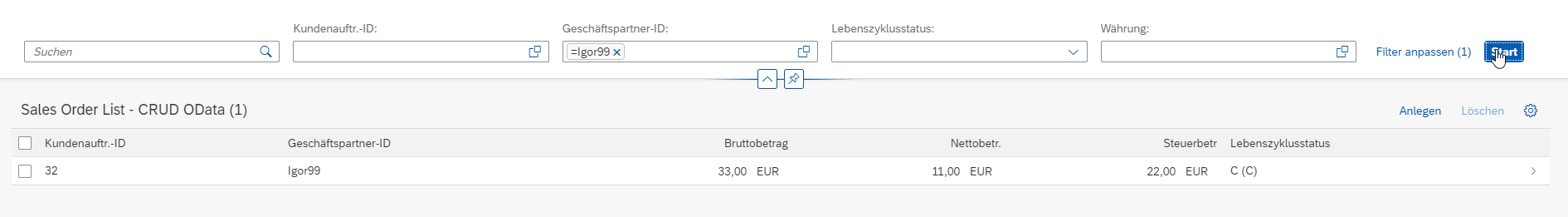
METHOD salesorderset\_delete\_entity.  
  
  DATA: ls\_so TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorder.  
  
  io\_tech\_request\_context->get\_converted\_keys(  
    IMPORTING  
      es\_key\_values = ls\_so ).  
  
  " Update the table rather than the CDS view itself.  
  DELETE FROM zproto\_so\_a WHERE salesorderid = ls\_so-salesorderid.  
  IF sy-subrc <> 0.  
  
    ROLLBACK WORK.  
  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    "Data does not exist.  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  005  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  ELSE.  
    " Delete also items  
    DELETE FROM zproto\_soi\_a WHERE salesorderid = ls\_so-salesorderid.  
  
  ENDIF.  
  
ENDMETHOD.

METHOD salesorderset\_get\_entity.  
  
  DATA: ls\_so TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorder.  
  
  io\_tech\_request\_context->get\_converted\_keys(  
    IMPORTING  
      es\_key\_values = ls\_so " Entity Key Values - converted  
  ).  
  
  SELECT SINGLE \*  
    FROM zddl\_i\_zproto\_so\_a  
    INTO @DATA(ls\_so\_db)  
    WHERE salesorderid = @ls\_so-salesorderid.  
  
  IF sy-subrc = 0.  
  
    MOVE-CORRESPONDING ls\_so\_db TO er\_entity.  
  
  ELSE.  
  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    "No Data Found  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  003  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  
  ENDIF.  
  
ENDMETHOD.

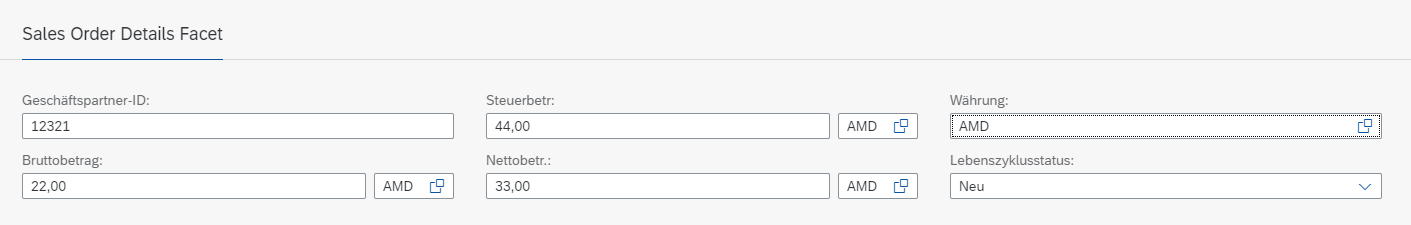
METHOD salesorderset\_get\_entityset.  
  
  TYPES: ty\_salesorderid      TYPE RANGE OF  zddl\_i\_zproto\_so\_a-salesorderid,  
         ty\_businesspartnerid TYPE RANGE OF  zddl\_i\_zproto\_so\_a-businesspartnerid,  
         ty\_currencycode      TYPE RANGE OF  zddl\_i\_zproto\_so\_a-currencycode,  
         ty\_grossamount       TYPE RANGE OF  zddl\_i\_zproto\_so\_a-grossamount,  
         ty\_netamount         TYPE RANGE OF  zddl\_i\_zproto\_so\_a-netamount,  
         ty\_taxamount         TYPE RANGE OF  zddl\_i\_zproto\_so\_a-taxamount,  
         ty\_lifecyclestatus   TYPE RANGE OF  zddl\_i\_zproto\_so\_a-lifecyclestatus.  
  
  DATA: lr\_salesorderid      TYPE ty\_salesorderid,  
        lr\_businesspartnerid TYPE ty\_businesspartnerid,  
        lr\_currencycode      TYPE ty\_currencycode,  
        lr\_grossamount       TYPE ty\_grossamount,  
        lr\_netamount         TYPE ty\_netamount,  
        lr\_taxamount         TYPE ty\_taxamount,  
        lr\_lifecyclestatus   TYPE ty\_lifecyclestatus.  
  
  DATA: lt\_sort TYPE abap\_sortorder\_tab,  
        ls\_sort TYPE LINE OF abap\_sortorder\_tab.  
  
  " URI filter <> $filter  
  LOOP AT it\_key\_tab INTO DATA(ls\_key\_tab).  
  
    TRANSLATE ls\_key\_tab-name TO UPPER CASE.  
  
    CASE ls\_key\_tab-name.  
      WHEN 'SALESORDERID'.  
        lr\_salesorderid[]      = VALUE ty\_salesorderid( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
        LOOP AT lr\_salesorderid ASSIGNING FIELD-SYMBOL(<fs\_so>).  
          <fs\_so>-low  =  |{ <fs\_so>-low  ALPHA = IN }|.  
          <fs\_so>-high =  |{ <fs\_so>-high ALPHA = IN }|.  
        ENDLOOP.  
      WHEN 'BUSINESSPARTNERID'.  
        lr\_businesspartnerid[] = VALUE ty\_businesspartnerid( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'CURRENCYCODE'.  
        lr\_currencycode[]      = VALUE ty\_currencycode( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'GROSSAMOUNT'.  
        lr\_grossamount[]       = VALUE ty\_grossamount( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'NETAMOUNT'.  
        lr\_netamount[]         = VALUE ty\_netamount( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'TAXAMOUNT'.  
        lr\_taxamount[]         = VALUE ty\_taxamount( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN 'LIFECYCLESTATUS'.  
        lr\_lifecyclestatus[]   = VALUE ty\_lifecyclestatus( ( sign = 'I' option = 'EQ' low = ls\_key\_tab-value ) ).  
      WHEN OTHERS.  
    ENDCASE.  
  ENDLOOP.  
  
  DATA(lr\_filter) = io\_tech\_request\_context->get\_filter( ).  
  IF lr\_filter IS BOUND.  
    DATA(lt\_filter\_so) = lr\_filter->get\_filter\_select\_options( ).  
  ENDIF.  
  
  "This is useful for dynamic queries.  
  DATA(lv\_sql) = io\_tech\_request\_context->get\_osql\_where\_clause\_convert( ).  
  DATA(lv\_sql2) = io\_tech\_request\_context->get\_osql\_where\_clause( ).  
  
  " $filter  
  LOOP AT lt\_filter\_so REFERENCE INTO DATA(lr\_filter\_so).  
    CASE lr\_filter\_so->property.  
      WHEN 'SALESORDERID'.  
        lr\_salesorderid[] = CORRESPONDING #( lr\_filter\_so->select\_options ).  
        LOOP AT lr\_salesorderid ASSIGNING <fs\_so>.  
          <fs\_so>-low  =  |{ <fs\_so>-low  ALPHA = IN }|.  
          <fs\_so>-high =  |{ <fs\_so>-high ALPHA = IN }|.  
        ENDLOOP.  
      WHEN 'BUSINESSPARTNERID'.  
        lr\_businesspartnerid[] = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'CURRENCYCODE'.  
        lr\_currencycode[]      = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'GROSSAMOUNT'.  
        lr\_grossamount[]       = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'NETAMOUNT'.  
        lr\_netamount[]         = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'TAXAMOUNT'.  
        lr\_taxamount[]         = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN 'LIFECYCLESTATUS'.  
        lr\_lifecyclestatus[]   = CORRESPONDING #( lr\_filter\_so->select\_options ).  
      WHEN OTHERS.  
    ENDCASE.  
  ENDLOOP.  
  
  SELECT salesorderid, businesspartnerid, currencycode, grossamount, netamount, taxamount, lifecyclestatus  
    FROM zddl\_i\_zproto\_so\_a  
    INTO TABLE @DATA(lt\_so)  
    WHERE salesorderid      IN @lr\_salesorderid  
      AND businesspartnerid IN @lr\_businesspartnerid  
      AND currencycode      IN @lr\_currencycode  
      AND grossamount       IN @lr\_grossamount  
      AND netamount         IN @lr\_netamount  
      AND taxamount         IN @lr\_taxamount  
      AND lifecyclestatus   IN @lr\_lifecyclestatus.  
  
  IF sy-subrc = 0.  
  
    " $orderby  
    DATA(lt\_orderby) = io\_tech\_request\_context->get\_orderby( ).  
    LOOP AT lt\_orderby INTO DATA(ls\_orderby).  
      ls\_sort-name = ls\_orderby-property.  
      IF ls\_orderby-order = 'desc'.  
        ls\_sort-descending = abap\_true.  
      ENDIF.  
      INSERT ls\_sort INTO TABLE lt\_sort.  
      CLEAR ls\_sort.  
    ENDLOOP.  
  
    SORT lt\_so BY (lt\_sort).  
  
    " $inlinecount  
    IF io\_tech\_request\_context->has\_inlinecount( ).  
      es\_response\_context-inlinecount = lines( lt\_so ).  
    ENDIF.  
    " $skip  
    IF is\_paging-skip IS NOT INITIAL.  
      DO is\_paging-skip TIMES.  
        DELETE lt\_so[] INDEX 1.  
      ENDDO.  
    ENDIF.  
    " $top  
    IF is\_paging-top IS NOT INITIAL.  
      LOOP AT lt\_so INTO DATA(ls\_so).  
        IF sy-tabix > is\_paging-top.  
          DELETE lt\_so INDEX sy-tabix.  
        ENDIF.  
      ENDLOOP.  
    ENDIF.  
  
    et\_entityset[] = CORRESPONDING #( lt\_so ).  
  
  ELSE.  
  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  003  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  
  ENDIF.  
  
ENDMETHOD.

METHOD salesorderset\_update\_entity.  
  
  DATA: ls\_so    TYPE zcl\_zso\_no\_cds\_mpc=>ts\_salesorder,  
        ls\_so\_db TYPE zproto\_so\_a.  
  
  io\_data\_provider->read\_entry\_data(  
    IMPORTING  
      es\_data = ls\_so ).  
  
  ls\_so\_db  = CORRESPONDING #( ls\_so ).  
  er\_entity = CORRESPONDING #( ls\_so ).  
  
  UPDATE zproto\_so\_a FROM ls\_so\_db.  
  IF sy-subrc <> 0.  
  
    DATA(lo\_message\_container) = /iwbep/if\_mgw\_conv\_srv\_runtime~get\_message\_container( ).  
    "Error during update  
    lo\_message\_container->add\_message(  
      EXPORTING  
        iv\_msg\_type               = 'E'  
        iv\_msg\_id                 = 'ZIMU'  
        iv\_msg\_number             =  006  ).  
  
    RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception  
      EXPORTING  
        message\_container = lo\_message\_container.  
  
  ENDIF.  
  
ENDMETHOD.

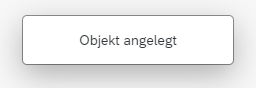
1. Service Activation /IWFND/MAINT\_SERVICE - Services aktivieren und verwalten
2. SAP UI5 Coding:  
     
   
3. Results:  
     
   

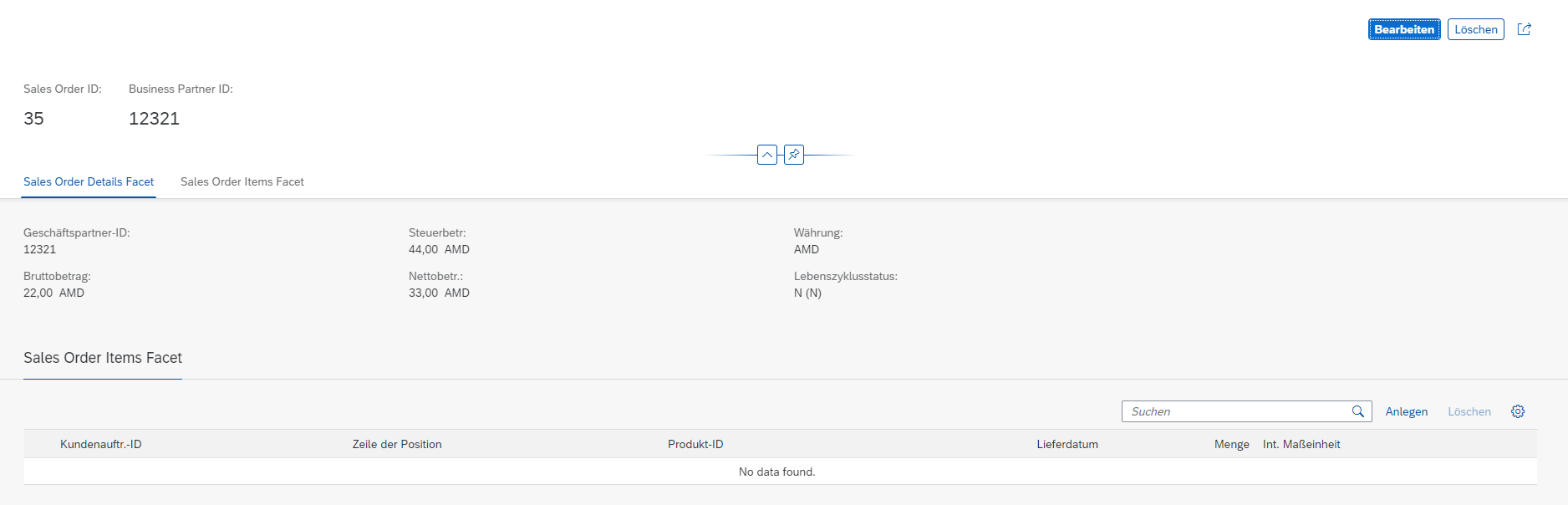


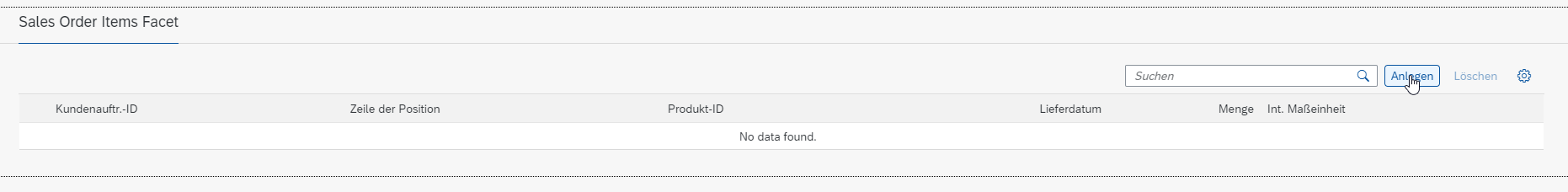


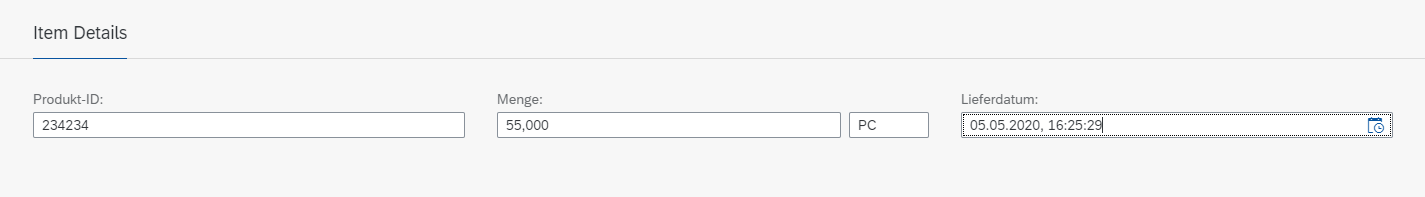




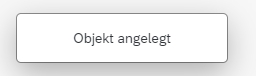














Checking the tables:

