****

**ABAP Development for SAP HANA**

[**https://open.sap.com/courses/a4h1**](https://open.sap.com/courses/a4h1)

**Source Code of Week 1**



Version 1 - 15.10.2014

Only Demo content

Without any warranty

**Contents**

[CDS DDL Sources 4](#_Toc401148562)

[ZDDLS\_CDS\_40\_ANNOTATION 5](#_Toc401148563)

[ABAP Classes 6](#_Toc401148564)

[ZCL\_DEMO\_PAID\_ON\_DATE 7](#_Toc401148565)

[ZCL\_DEMO\_PAID\_ON\_DATE\_AMDP 11](#_Toc401148566)

[ABAP Programs 13](#_Toc401148567)

[ZR\_HELLO\_WORLD 14](#_Toc401148568)

[ZR\_EPM\_SET\_INVOICE\_PAID 15](#_Toc401148569)

[ZR\_PAID\_ON\_DATE 20](#_Toc401148570)

[ZR\_PAID\_ON\_DATE\_AMDP 21](#_Toc401148571)

**CDS DDL Sources**

**ZDDLS\_CDS\_40\_ANNOTATION**

@AbapCatalog.sqlViewName: 'ZDDLS\_CDS\_40'

@ClientDependent: true

@AbapCatalog.Buffering.status: #SWITCHED\_OFF

**define** **view** zcdsv\_annotation\_simple **as** **select** **from** snwd\_so

**{**

**key** so\_id **as** customer\_id**,**

@Semantics.currencyCode: true

currency\_code**,**

@Semantics.amount.currencyCode: 'currency\_code'

gross\_amount

**}**

**ABAP Classes**

**ZCL\_DEMO\_PAID\_ON\_DATE**

CLASS zcl\_demo\_paid\_on\_date DEFINITION

PUBLIC

CREATE PUBLIC .

PUBLIC SECTION.

TYPES:

BEGIN OF ty\_invoice\_header,

invoice\_guid TYPE snwd\_so\_inv\_head-node\_key,

created\_at TYPE snwd\_so\_inv\_head-created\_at,

paid\_at TYPE snwd\_so\_inv\_head-changed\_at,

buyer\_guid TYPE snwd\_so\_inv\_head-buyer\_guid,

END OF ty\_invoice\_header .

TYPES:

BEGIN OF ty\_invoice\_item,

item\_guid TYPE snwd\_so\_inv\_item-node\_key,

invoice\_guid TYPE snwd\_so\_inv\_head-node\_key,

product\_guid TYPE snwd\_so\_inv\_item-product\_guid,

gross\_amount TYPE snwd\_so\_inv\_item-gross\_amount,

currency\_code TYPE snwd\_so\_inv\_item-currency\_code,

END OF ty\_invoice\_item .

TYPES:

BEGIN OF ty\_customer\_info,

customer\_guid TYPE snwd\_bpa-node\_key,

customer\_id TYPE snwd\_bpa-bp\_id,

customer\_name TYPE snwd\_bpa-company\_name,

country TYPE snwd\_ad-country,

postal\_code TYPE snwd\_ad-postal\_code,

city TYPE snwd\_ad-city,

END OF ty\_customer\_info .

TYPES:

tt\_invoice\_header TYPE STANDARD TABLE OF ty\_invoice\_header WITH KEY invoice\_guid .

TYPES:

tt\_invoice\_item TYPE STANDARD TABLE OF ty\_invoice\_item .

TYPES:

tt\_customer\_info TYPE STANDARD TABLE OF ty\_customer\_info .

METHODS paid\_on\_date

IMPORTING

VALUE(iv\_payment\_date) TYPE d

EXPORTING

VALUE(et\_invoice\_header) TYPE tt\_invoice\_header

VALUE(et\_invoice\_item) TYPE tt\_invoice\_item

VALUE(et\_customer\_info) TYPE tt\_customer\_info .

ENDCLASS.

CLASS zcl\_demo\_paid\_on\_date IMPLEMENTATION.

METHOD paid\_on\_date.

"! selection of invoices paid on a specified date

"! plus business partner and product information

DATA ls\_invoice\_head TYPE ty\_invoice\_header.

DATA lt\_invoice\_item TYPE tt\_invoice\_item.

DATA lt\_customer\_info TYPE tt\_customer\_info.

DATA lv\_payment\_date\_min TYPE timestamp.

DATA lv\_payment\_date\_max TYPE timestamp.

CONVERT DATE iv\_payment\_date TIME '0000' INTO TIME STAMP lv\_payment\_date\_min TIME ZONE 'UTC'.

CONVERT DATE iv\_payment\_date TIME '2359' INTO TIME STAMP lv\_payment\_date\_max TIME ZONE 'UTC'.

" First we retrieve all invoice header

" which were paid on the requested date

SELECT

node\_key AS invoice\_guid

created\_at AS created\_at

changed\_at AS paid\_at

buyer\_guid

FROM

snwd\_so\_inv\_head

INTO ls\_invoice\_head

WHERE

payment\_status = 'P'

AND changed\_at BETWEEN lv\_payment\_date\_min AND lv\_payment\_date\_max.

CLEAR lt\_invoice\_item.

CLEAR lt\_customer\_info.

"get items of invoice

SELECT

node\_key AS item\_guid

parent\_key AS invoice\_guid

product\_guid

gross\_amount

currency\_code

FROM snwd\_so\_inv\_item

INTO TABLE lt\_invoice\_item

WHERE parent\_key = ls\_invoice\_head-invoice\_guid.

"get information about the customers

SELECT

bpa~node\_key AS customer\_guid

bpa~bp\_id AS customer\_id

bpa~company\_name AS customer\_name

ad~country

ad~postal\_code

ad~city

FROM snwd\_bpa AS bpa

JOIN snwd\_ad AS ad ON ad~node\_key = bpa~address\_guid

INTO TABLE lt\_customer\_info

WHERE bpa~node\_key = ls\_invoice\_head-buyer\_guid.

APPEND ls\_invoice\_head TO et\_invoice\_header.

APPEND LINES OF lt\_invoice\_item TO et\_invoice\_item.

APPEND LINES OF lt\_customer\_info TO et\_customer\_info.

ENDSELECT.

"remove duplicates

SORT et\_customer\_info BY customer\_name.

DELETE ADJACENT DUPLICATES FROM et\_customer\_info.

ENDMETHOD.

ENDCLASS.

**ZCL\_DEMO\_PAID\_ON\_DATE\_AMDP**

CLASS zcl\_demo\_paid\_on\_date\_amdp DEFINITION

PUBLIC

INHERITING FROM zcl\_demo\_paid\_on\_date

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES if\_amdp\_marker\_hdb.

METHODS paid\_on\_date REDEFINITION.

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS zcl\_demo\_paid\_on\_date\_amdp IMPLEMENTATION.

METHOD paid\_on\_date BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY

USING snwd\_so\_inv\_head snwd\_so\_inv\_item snwd\_bpa snwd\_ad.

--sql script

-- VALUE(et\_invoice\_header) TYPE tt\_invoice\_header

-- VALUE(et\_invoice\_item) TYPE tt\_invoice\_item

-- VALUE(et\_customer\_info) TYPE tt\_customer\_info .

et\_invoice\_header = SELECT

node\_key AS invoice\_guid,

created\_at AS created\_at,

changed\_at AS paid\_at,

buyer\_guid

FROM

snwd\_so\_inv\_head

WHERE

payment\_status = 'P'

AND LEFT(changed\_at,8) = :iv\_payment\_date;

et\_invoice\_item =

SELECT

node\_key AS item\_guid,

parent\_key AS invoice\_guid,

product\_guid,

gross\_amount,

currency\_code

FROM snwd\_so\_inv\_item

WHERE parent\_key in ( select invoice\_guid from :et\_invoice\_header );

--get information about the customers

et\_customer\_info = SELECT DISTINCT

bpa.node\_key AS customer\_guid,

bpa.bp\_id AS customer\_id,

bpa.company\_name AS customer\_name,

ad.country,

ad.postal\_code,

ad.city

FROM snwd\_bpa AS bpa

JOIN snwd\_ad AS ad ON ad.node\_key = bpa.address\_guid

WHERE bpa.node\_key in ( select buyer\_guid from :et\_invoice\_header )

ORDER BY company\_name;

ENDMETHOD.

ENDCLASS.

**ABAP Programs**

**ZR\_HELLO\_WORLD**

REPORT zr\_hello\_world.

DATA lv\_string TYPE string.

DATA lv\_name TYPE sy-uname.

lv\_name = cl\_abap\_syst=>get\_user\_name( ).

lv\_string = |Hello { lv\_name }, welcome to the ABAP Development for SAP HANA!|.

WRITE: lv\_string.

**ZR\_EPM\_SET\_INVOICE\_PAID**

REPORT zr\_epm\_set\_invoice\_paid.

CLASS lcl\_set\_invoice\_paid DEFINITION CREATE PRIVATE.

PUBLIC SECTION.

CLASS-DATA mv\_test\_mode TYPE abap\_bool.

CLASS-METHODS execute

IMPORTING

iv\_bupa\_id TYPE snwd\_bpa-bp\_id

iv\_count TYPE i

EXPORTING

ev\_open TYPE i

et\_so TYPE if\_epm\_so\_header=>tt\_node\_data

RAISING

cx\_epm\_api\_exception

cx\_epm\_system\_exception.

ENDCLASS.

CLASS lcl\_set\_invoice\_paid IMPLEMENTATION.

METHOD execute.

DATA(mo\_message\_buffer) = cl\_epm\_service\_facade=>get\_message\_buffer( ).

DATA mo\_invoice TYPE REF TO if\_epm\_so\_invoice.

mo\_invoice ?= cl\_epm\_service\_facade=>get\_bo( if\_epm\_so\_invoice=>gc\_bo\_name ).

" read the invoices for this bupa

mo\_invoice->if\_epm\_so\_invoice\_header~query\_by\_header(

EXPORTING it\_sel\_par\_buyer\_ids = VALUE #( ( sign = 'I' option = 'EQ' low = iv\_bupa\_id ) )

iv\_max\_rows = 99999

IMPORTING et\_data = DATA(lt\_invoices) ).

DATA lt\_paid\_invoice LIKE lt\_invoices.

" find some unpaid invoices

DATA(lv\_count) = iv\_count.

LOOP AT lt\_invoices ASSIGNING FIELD-SYMBOL(<ls\_invoice>).

CHECK lv\_count > 0.

DATA lt\_invoice\_key TYPE STANDARD TABLE OF snwd\_so-node\_key.

IF <ls\_invoice>-payment\_status = space.

APPEND <ls\_invoice>-node\_key TO lt\_invoice\_key.

INSERT <ls\_invoice> INTO TABLE lt\_paid\_invoice.

lv\_count = lv\_count - 1.

ENDIF.

ENDLOOP.

ev\_open = lv\_count.

IF lv\_count > 0.

RETURN.

ENDIF.

" mark selected as paid

mo\_invoice->if\_epm\_so\_invoice\_header~action\_mark\_as\_paid(

EXPORTING

it\_node\_keys = lt\_invoice\_key

ii\_message\_buffer = mo\_message\_buffer

IMPORTING

et\_node\_key\_info = DATA(lt\_node\_key\_info)

).

DATA(lt\_messages) = mo\_message\_buffer->get\_messages( ).

LOOP AT lt\_messages ASSIGNING FIELD-SYMBOL(<ls\_message>).

DATA(lv\_text) = <ls\_message>->to\_string( ).

WRITE: / lv\_text.

ENDLOOP.

cl\_epm\_service\_facade=>save(

iv\_suppress\_commit = mv\_test\_mode

ii\_message\_buffer = mo\_message\_buffer ).

lt\_messages = mo\_message\_buffer->get\_messages( ).

LOOP AT lt\_messages ASSIGNING <ls\_message>.

lv\_text = <ls\_message>->to\_string( ).

WRITE: / lv\_text.

ENDLOOP.

mo\_invoice->if\_epm\_so\_invoice\_header~navigate\_to\_sales\_order(

EXPORTING it\_source\_node\_keys = lt\_invoice\_key

IMPORTING et\_data = et\_so

).

ENDMETHOD.

ENDCLASS.

CLASS ltc\_invoice\_paid DEFINITION CREATE PRIVATE

FOR TESTING

DURATION MEDIUM

RISK LEVEL DANGEROUS.

PRIVATE SECTION.

METHODS setup.

METHODS teardown.

METHODS do\_test FOR TESTING RAISING cx\_epm\_api\_exception cx\_epm\_system\_exception.

ENDCLASS.

CLASS ltc\_invoice\_paid IMPLEMENTATION.

METHOD setup.

lcl\_set\_invoice\_paid=>mv\_test\_mode = abap\_true.

ENDMETHOD.

METHOD do\_test.

DATA lv\_bpa TYPE snwd\_bpa-bp\_id.

SELECT SINGLE bpa~bp\_id

FROM snwd\_bpa AS bpa

INNER JOIN snwd\_so AS so ON so~buyer\_guid = bpa~node\_key

INNER JOIN snwd\_so\_inv\_head AS head ON head~so\_guid = so~node\_key

INTO lv\_bpa

WHERE head~payment\_status = ''.

cl\_abap\_unit\_assert=>assert\_subrc(

msg = 'no corresponding data in the DB' ##no\_text

level = if\_aunit\_constants=>tolerable

quit = if\_aunit\_constants=>method ).

lcl\_set\_invoice\_paid=>execute(

EXPORTING

iv\_bupa\_id = lv\_bpa

iv\_count = 1

IMPORTING

ev\_open = DATA(lv\_open)

et\_so = DATA(lt\_so)

).

cl\_abap\_unit\_assert=>assert\_initial( act = lv\_open ).

cl\_abap\_unit\_assert=>assert\_not\_initial( act = lt\_so ).

ENDMETHOD.

METHOD teardown.

lcl\_set\_invoice\_paid=>mv\_test\_mode = abap\_false.

ROLLBACK WORK.

ENDMETHOD.

ENDCLASS.

PARAMETER bupa TYPE snwd\_bpa-bp\_id.

PARAMETER num TYPE i DEFAULT 1.

START-OF-SELECTION.

lcl\_set\_invoice\_paid=>execute(

EXPORTING

iv\_bupa\_id = bupa

iv\_count = num

IMPORTING

ev\_open = DATA(lv\_open)

et\_so = DATA(lt\_so)

).

IF lv\_open > 0.

WRITE: / 'not enough unpaid invoices available'(001).

ELSE.

WRITE: / 'sales orders marked as paid:'(002).

LOOP AT lt\_so ASSIGNING FIELD-SYMBOL(<ls\_paid>).

WRITE: / <ls\_paid>-so\_id, <ls\_paid>-gross\_amount, <ls\_paid>-currency\_code.

ENDLOOP.

ENDIF.

**ZR\_PAID\_ON\_DATE**

PROGRAM zr\_paid\_on\_date.

DATA p\_date TYPE d VALUE '20140912'.

DATA(lo\_timer) = cl\_abap\_runtime=>create\_hr\_timer( ).

DATA(lo\_info\_list) = NEW zcl\_demo\_paid\_on\_date( ).

DATA(t1) = lo\_timer->get\_runtime( ).

lo\_info\_list->paid\_on\_date(

EXPORTING

iv\_payment\_date = p\_date

IMPORTING

et\_invoice\_header = DATA(lt\_invoice\_head)

et\_invoice\_item = DATA(lt\_invoice\_item)

et\_customer\_info = DATA(lt\_customer\_info) ).

DATA(t2) = lo\_timer->get\_runtime( ).

DATA(elapsed\_time) = ( t2 - t1 ) / 1000.

cl\_demo\_output=>next\_section( title = |Runtime (ABAP): { elapsed\_time } ms.| ).

cl\_demo\_output=>write\_data( name = 'Customer Info' value = lt\_customer\_info ).

cl\_demo\_output=>display( ).

**ZR\_PAID\_ON\_DATE\_AMDP**

PROGRAM zr\_paid\_on\_date\_amdp.

DATA p\_date TYPE d VALUE '20140912'.

DATA(lo\_timer) = cl\_abap\_runtime=>create\_hr\_timer( ).

DATA(lo\_info\_list) = NEW zcl\_demo\_paid\_on\_date\_amdp( ).

DATA(t1) = lo\_timer->get\_runtime( ).

lo\_info\_list->paid\_on\_date(

EXPORTING

iv\_payment\_date = p\_date

IMPORTING

et\_invoice\_header = DATA(lt\_invoice\_head)

et\_invoice\_item = DATA(lt\_invoice\_item)

et\_customer\_info = DATA(lt\_customer\_info) ).

DATA(t2) = lo\_timer->get\_runtime( ).

DATA(elapsed\_time) = ( t2 - t1 ) / 1000.

cl\_demo\_output=>next\_section( title = |Runtime (AMDP): { elapsed\_time } ms.| ).

cl\_demo\_output=>write\_data( name = 'Customer Info' value = lt\_customer\_info ).

cl\_demo\_output=>display( ).