CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC

PUBLIC SECTION.

INTERFACES: if\_andp\_marker\_hdb.

• ROWCOUNT

CLASS-METHODS: get\_info EXPORTING VALUE (et\_spfli) TYPE spfl tab RAISING cx\_amdp\_error.

ET SPFLI

LT CARRID

LT CARRID2

ENDCLASS.

CLASS zel\_amdp\_explore IMPLEMENTATION.

METHOD get info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT

LT CARRID ALL

LT CARRID DISTINCT

USING spfli scarr.

1t\_carrid1 = SELECT carrid FROM scarr WHERE mandt = session\_context( 'CLIENT' );

1t\_carrid2 = SELECT carrid FROM scarr WHERE mandt = session\_context( 'CLIENT' );

lt\_carrid distinct = SELECT FROM :1t\_carridl

UNION

SELECT FROM :1t\_carrid2;

it\_carrid\_all =SELECT FROM:1t\_carridi

UNION ALL

SELECT FROM:1t\_carrid2;

et spfli= SELECT FROM spfli

WHERE mandt =

session\_context(

'CLIENT' );

TABLE [38]

ENDMETHOD.

1. Round off Function:

**CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC .**

**PUBLIC SECTION.**

**INTERFACES: if\_amdp\_marker\_hdb.**

**CLASS-METHODS: get\_info EXPORTING VALUE(et\_spfli) TYPE spfli\_tab**

**RAISING cx\_amdp\_error.**

**ENDCLASS.**

**CLASS zcl\_amdp\_explore IMPLEMENTATION.**

**METHOD get\_info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT USING spfli.**

**— Syntax- ROUND(<number> [, <position> [, <rounding\_mode>]])**

**lt\_v111 = SELECT ROUND (111.38, 1, ROUND\_HALF\_UP ) “val” FROM DUMMY; — (111.40)**

**lt\_v112 = SELECT ROUND (111.38, 1, ROUND\_HALF\_DOWN ) “val” FROM DUMMY; — (111.40)**

**lt\_v113 = SELECT ROUND (111.38, 1, ROUND\_HALF\_EVEN ) “val” FROM DUMMY; — (111.40)**

**lt\_v114 = SELECT ROUND (111.38, 1, ROUND\_UP ) “val” FROM DUMMY; — (111.40)**

**lt\_v115 = SELECT ROUND (111.38, 1, ROUND\_DOWN ) “val” FROM DUMMY; — (111.30)**

**lt\_v116 = SELECT ROUND (111.38, 1, ROUND\_CEILING ) “val” FROM DUMMY; — (111.40)**

**lt\_v117 = SELECT ROUND (111.38, 1, ROUND\_FLOOR ) “val” FROM DUMMY; — (111.30)**

**lt\_v118 = SELECT ROUND (-111.38, 1, ROUND\_UP ) “val” FROM DUMMY; — (-111.40)**

**lt\_v119 = SELECT ROUND (-111.38, 1, ROUND\_DOWN ) “val” FROM DUMMY; — (-111.30)**

**lt\_v120 = SELECT ROUND (-111.38, 1, ROUND\_CEILING ) “val” FROM DUMMY; — (-111.30)**

**lt\_v121 = SELECT ROUND (-111.38, 1, ROUND\_FLOOR ) “val” FROM DUMMY; — (-111.40)**

**lt\_v221 = SELECT ROUND (111.35, 1, ROUND\_HALF\_UP ) “val” FROM DUMMY; — (111.40)**

**lt\_v222 = SELECT ROUND (111.35, 1, ROUND\_HALF\_DOWN ) “val” FROM DUMMY; — (111.30)**

**lt\_v223 = SELECT ROUND (111.35, 1, ROUND\_HALF\_EVEN ) “val” FROM DUMMY; — (111.40)**

**lt\_v224 = SELECT ROUND (111.35, 1, ROUND\_UP ) “val” FROM DUMMY; — (111.40)**

**lt\_v225 = SELECT ROUND (111.35, 1, ROUND\_DOWN ) “val” FROM DUMMY; — (111.30)**

**lt\_v226 = SELECT ROUND (111.35, 1, ROUND\_CEILING ) “val” FROM DUMMY; — (111.40)**

**lt\_v227 = SELECT ROUND (111.35, 1, ROUND\_FLOOR ) “val” FROM DUMMY; — (111.30)**

**lt\_v228 = SELECT ROUND (-111.35, 1, ROUND\_UP ) “val” FROM DUMMY; — (-111.40)**

**lt\_v229 = SELECT ROUND (-111.35, 1, ROUND\_DOWN ) “val” FROM DUMMY; — (-111.30)**

**lt\_v230 = SELECT ROUND (-111.35, 1, ROUND\_CEILING ) “val” FROM DUMMY; — (-111.30)**

**lt\_v231 = SELECT ROUND (-111.35, 1, ROUND\_FLOOR ) “val” FROM DUMMY; — (-111.40)**

**lt\_v331 = SELECT ROUND (111.32, 1, ROUND\_HALF\_UP ) “val” FROM DUMMY; — (111.30)**

**lt\_v332 = SELECT ROUND (111.32, 1, ROUND\_HALF\_DOWN ) “val” FROM DUMMY; — (111.30)**

**lt\_v333 = SELECT ROUND (111.32, 1, ROUND\_HALF\_EVEN ) “val” FROM DUMMY; — (111.30)**

**lt\_v334 = SELECT ROUND (111.32, 1, ROUND\_UP ) “val” FROM DUMMY; — (111.40)**

**lt\_v335 = SELECT ROUND (111.32, 1, ROUND\_DOWN ) “val” FROM DUMMY; — (111.30)**

**lt\_v336 = SELECT ROUND (111.32, 1, ROUND\_CEILING ) “val” FROM DUMMY; — (111.40)**

**lt\_v337 = SELECT ROUND (111.32, 1, ROUND\_FLOOR ) “val” FROM DUMMY; — (111.30)**

**lt\_v338 = SELECT ROUND (-111.32, 1, ROUND\_UP ) “val” FROM DUMMY; — (-111.40)**

**lt\_v339 = SELECT ROUND (-111.32, 1, ROUND\_DOWN ) “val” FROM DUMMY; — (-111.30)**

**lt\_v340 = SELECT ROUND (-111.32, 1, ROUND\_CEILING ) “val” FROM DUMMY; — (-111.30)**

**lt\_v341 = SELECT ROUND (-111.32, 1, ROUND\_FLOOR ) “val” FROM DUMMY; — (-111.40)**

**lt\_v441 = SELECT ROUND (115.38, 1 ) “val” FROM DUMMY; — (115.40)**

**lt\_v442 = SELECT ROUND (115.38, -1 ) “val” FROM DUMMY; — (120.00)**

**et\_spfli = SELECT \* FROM spfli WHERE mandt = session\_context( ‘CLIENT’ );**

**ENDMETHOD.**

**ENDCLASS.**

1. UUID:

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: get\_info EXPORTING VALUE (et spfli) TYPE spfli\_tab

ENDCLASS.

RAISING cx\_amdp\_error.

CLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD get\_info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT USING spfli.

-SYSUUIS Function- generates a new universally unique identifier lt spfli SELECT carrid,

connid,

sysuuid as uuid

FROM spfli WHERE mandt = session\_context( 'CLIENT' ); et\_spfli= SELECT \* FROM spfli WHERE mandt = session\_context( 'CLIENT' );

ENDMETHOD.

ENDCLASS.

1. Trace in ADT:

eCLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC. PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb. INTERFACES: if\_oo\_adt\_classrun.

CLASS-METHODS: get\_info EXPORTING VALUE (et\_spfli) TYPE spfli\_tab

ENDCLASS.

RAISING cx\_amdp\_error.

CLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD get\_info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT USING

spfli.

et\_spfli SELECT FROM spfli WHERE mandt session\_context( 'CLIENT') =

ENDMETHOD.

METHOD if oo\_adt\_classrun-main.

AND carrid = 'LH';

get\_info(IMPORTING et spfli= DATA (lt\_spfli) ). out->write( EXPORTING data = lt\_spfli ).

ENDMETHOD.

ENDCLASS.

\*\*\*\* Trace with compiler \*\*\*\*:

CLASS zel\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC.

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb. CLASS-METHODS: get\_info EXPORTING VALUE (et\_spfli) type spfli tab RAISING cx\_amdp\_error.

ENDCLASS.

CLASS zel\_amdp\_explore IMPLEMENTATION. METHOD get\_info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT USING spfli.

et\_spfli= SELECT FROM spfli WHERE mandt = session\_context( 'CLIENT')

AND carrid 'LH';

It spfli=SELECT FROM spfli WHERE mandt = session\_context( 'CLIENT');

et\_spfli =SELECT \* FROM :1t\_spfli WHERE carrid 'LH';

lt\_spfli= SELECT FROM spfli WHERE mandt = session\_context( 'CLIENT' )

WITH HINT (NO\_INLINE);

et spfli =SELECT FROM :1t\_spfli WHERE carrid = 'LH';

ENDMETHOD.

ENDCLASS.

\*\*\*\* upper and lower function \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC.

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb. CLASS-METHODS: get\_info.

#ROWCOUNT

CURRENT OBJE SAPHANADB N

0

ENDCLASS.

LV\_LVAL

abcd

eCLASS zcl\_amdp\_explore IMPLEMENTATION.

LV UVAL

LV\_VAL1

LV\_VAL2

XYZ

ABCD

ABCD

METHOD get\_info BY DATABASE PROCEDURE

FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY.

LV VAL3

xyz

E

V

V

V

V

lv\_lval

VARCHAR( 4 )

DEFAULT 'abcd';

DECLARE

DECLARE lv\_uval

DECLARE

VARCHAR(4) DEFAULT 'XYZ'; VARCHAR( 4 );

DECLARE

lv\_vall

lv\_val2 VARCHAR ( 4 );

DECLARE lv\_val3 VARCHAR( 4 );

lv\_val1 = upper

( lv\_lval );

lv\_val2 = ucase

( lv\_lval );

lv\_val3

= lower (

lv\_uval );

Name

ENDMETHOD.

ENDCLASS.

\*\*\*\* IFNULL && NULIFF \*\*\*\*

IFNULL-

CLASS zel amdp\_explore IMPLEMENTATION.

METHOD get info BY DATABASE PROCEDURE

FOR HDB LANGUAGE SOLSCRIPT OPTIONS READ-ONLY.

DECLARE 1v vall INTEGER/

DECLARE 1v val2 INTEGER DEFAULT 10:

EV WALT

LV VAL2

DECLARE 1v val3 INTEGER

DECLARE lv\_val4 INTEGER

EV VALS

LV VALA

IFNULL() returns first operand if the value is not null,

else returns the second argument

lv\_val3 IFNULL( lv\_vall, 1 17

lv\_val4 IFNULL( lv\_val2, 0);

ENDNETHOD

ENDCLASS.

NULLIF-

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC

PUBLIC SECTION.

CURRENT OBJE

CURRENT OBIE

ROWCOUNT

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: get\_info.

WV CREST

LV CRES2

ENDCLASS.

CLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD get info

BY DATABASE PROCEDURE

FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY.

LV CVAL2

LV CVALT

IV CVAL3

DECLARE 1v cvall VARCHAR(4)

DEFAULT 'abcd';

DECLARE 1v cval2 VARCHAR(4) DEFAULT 'abcd';

DECLARE 1v cval3 VARCHAR(4) DEFAULT 'pqrs';

DECLARE 1v cres1 VARCHAR( 4 );

DECLARE 1v\_cres2 VARCHAR( 4 );

NULLIF() retruns NULL if the values else returns the first argument

are same

lv\_cresi = NULLIF( lv\_evall, lv\_eval2 );

1v cvall, lv\_cval3 ); lv\_cres2 NULLIF( =

ENDMETHOD.

ENDCLASS.

\*\*\*\* MAP Function \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC.

CLASS-METHODS: get\_info.

ENDCLASS.

CLASS zcl\_amdp\_explore IMPLEMENTATION. get\_info BY DATABASE PROCEDURE

METHOD

FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY.

DECLARE lv\_date DATE;

DECLARE lv\_dats "SABAP. TYPE ( sy-datum )";

DECLARE lv\_dayname "$ABAP.TYPE( string )";

DECLARE lv\_number INTEGER;

lv\_date = CURRENT DATE: current date

lv\_dats = TO\_DATS (CURRENT\_DATE); covert current date to YYYYMMDD type lv\_dayname DAYNAME (CURRENT\_DATE); convert date to day name

MAP Fucntics Its like IF-THEN-ELSE

lv\_number = MAP DAYNAME (CURRENT\_DATE), --argument MONDAY, 1, if-then

'TUESDAY', 2, if-then

WEDNESDAY', 3, if-then

THURSDAY', 4, if-then

'FRIDAY',

5,

if-then

'SATURDAY', 6, if-then

'SUNDAY', 7, if-then

0);

else

SAF

DE

RECENT POSTS

• Internal Table w

• What is an API?

• Exposing service

• EXPOSE SERVIC

• EXPOSE Parame

Exposing service

• Test APIs throug

• Find User who ra

Email Address

BLOG STATS

ENDMETHOD.

ENDCLASS.

\*\*\*\* Session context \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC. PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: set\_input IMPORTING VALUE (iv\_carrid) TYPE s\_carr\_id.

CLASS-METHODS: unset\_input.

CLASS-METHODS: get\_info EXPORTING VALUE (et\_spfli) TYPE spfli\_tab. ENDCLASS.

CLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD set\_input BY DATABASE PROCEDURE

FOR HDB LANGUAGE SQLSCRIPT. SET 22 CARRID':iv\_carrid;

ENDMETHOD.

METHOD unset\_input BY DATABASE PROCEDURE

FOR HDB LANGUAGE SQLSCRIPT.

UNSET ZZ CARRID';

ENDMETHOD.

METHOD get\_info BY DATABASE PROCEDURE

FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY USING spfli. DECLARE lv\_carrid "SABAP.TYPE( s\_carr\_id )";

lv\_carrid = SESSION\_CONTEXT( 'ZZ\_CARRID' );

et\_spfli= SELECT FROM SPFLI WHERE mandt = SESSION\_CONTEXT('CLIENT')

AND carrid = lv\_carrid;

ENDMETHOD.

ENDCLASS.

\*\*\*\* Default Null \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb. CLASS-METHODS: get\_info EXPORTING VALUE (et\_spfli) TYPE spfli\_tab. ENDCLASS.

@CLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD get\_info BY DATABASE PROCEDURE

FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY USING scarr spfli.

DECLARE lv\_Carridi VARCHAR( 3 );

DECLARE lv\_Carrid2 VARCHAR( 3 );

lt\_scarr = SELECT \* FROM scarr WHERE mandt session\_context( 'CLIENT' );.

when selecting single field from DB table or internal table if the

record not found then reulsts in DUMP SELECT carrid INTO lv\_Carrid1 FROM Scarr WHERE mandt = session\_context( 'CLIENT' )

AND carrid = 'AA';

SELECT carrid INTO lv\_Carrid2 FROM :1t\_Scarr WHERE carrid = 'AA';

A

et\_spfli= SELECT \* FROM spfli WHERE mandt session\_context( 'CLIENT' );

ENDMETHOD.

ENDCLASS.

\*\*\*\* Exit Handler \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: get\_info EXPORTING VALUE (et\_spfli) TYPE spfli\_tab. ENDCLASS.

eCLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD get\_info BY DATABASE PROCEDURE

FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY USING scarr spfli.

DECLARE lv\_carrid1 VARCHAR( 3 );

lt\_scarr = SELECT FROM scarr WHERE mandt = session\_context( 'CLIENT' ); SELECT carrid INTO lv\_carrid1 FROM :1t\_Scarr WHERE carrid = 'XY';

et\_spfli= SELECT \* FROM spfli WHERE mandt session\_context( 'CLIENT' );

ENDMETHOD.

ENDCLASS.

\*\*\*\* signal and resignal \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC. PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: get\_info IMPORTING VALUE (iv\_num) TYPE i EXPORTING VALUE (et\_spfli) TYPE spfli\_tab.

ENDCLASS.

CLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD get\_info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY USING spfli. user defined error code: 10000 to 19999

DECLARE invalid\_input CONDITION FOR SQL\_ERROR\_CODE 10000;

IF iv\_num = 0 THEN

SIGNAL invalid\_input; raise user define exception

END IF;

et\_spfli= SELECT \* FROM spfli WHERE mandt session\_context( 'CLIENT' );

ENDMETHOD.

ENDCLASS.

\*\*\*\* Autonomous Transaction \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC. PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: update flight IMPORTING VALUE (iv\_carrid) TYPE s\_carr\_id VALUE (iv\_url) TYPE S CARRURL.

ENDCLASS.

CLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD update\_flight BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT

USING scarr.

AUTOMATIC TRANSACTION has an implicit COMMIT and independent of main transaction

BEGIN AUTONOMOUS TRANSACTION

UPDATE scarr SET url = iv\_url WHERE mandt = SESSION\_CONTEXT('CLIENT')

AND carrid :iv\_carrid;

END;

ENDMETHOD.

ENDCLASS.

\*\*\*\* Cursor \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC.

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: get\_info IMPORTING VALUE (iv\_carrid) TYPE s\_carr\_id.

ENDCLASS.

eCLASS zcl\_amdp\_explore IMPLEMENTATION.

METHOD get\_info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY USING scarr.

DECLARE lv\_carrid SABAP.type ( s\_Carr\_id )";

DECLARE lv\_carrname "SABAP.type(s\_Carrname )";

DECLARE lv\_currcode "SABAP.type( s\_Currcode )";

DECLARE CURSOR c\_flight ( ic\_carrid "SABAP.type(s\_Carr\_id )") FOR SELECT carrid, carrname, currcode FROM scarr

WHERE carrid = :ic\_carrid;

OPEN C\_FLIGHT ( 'AA' );

FETCH C\_FLIGHT INTO lv\_carrid, lv\_carrname, lv\_currcode:

C\_FLIGHT( 'AB' );

FETCH C\_FLIGHT INTO lv\_carrid, lv\_carrname, lv\_currcode:

OPEN C\_FLIGHT ( iv\_carrid );

FETCH C\_FLIGHT INTO lv\_carrid, lv\_carrname, lv\_currcode:

CLOSE C FLIGHT;

ENDMETHOD.

ENDCLASS.

\*\*\*\* Operation 1 \*\*\*\*

CLASS zcl\_amdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC.

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

CLASS-METHODS: get\_info

EXPORTING VALUE (et\_scarr) TYPE ty\_scarr

RAISING cx\_amdp\_error.

ENDCLASS.

CLASS zc1\_amdp\_explore IMPLEMENTATION.

METHOD get\_info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY USING scarr.

Declaration of Internal Table

DECLARE 1t scarr1 TABLE (carrid

"SABAP-type(s\_carr\_id )",

carrname "$ABAP.TYPE( s\_carrname )",

currcode "$ABAP.TYPE(s\_currcode )");

Declar Int Table with ref to another Int Table DECLARE 1t scarr2 table like :lt\_scarri:

DECLARE IV index INTEGER;

DECLARE lv\_line INTEGER;

It scarri= SELECT carrid, carrname, currcode

FROM scarr WHERE mandt = Session\_context( 'CLIENT' );

lv\_index= 1;

lv\_line = record\_count : 1t\_Scarrl); No. of records in an Int Table

if lv\_line <> 0 THEN

WHILE lv index BETWEEN 1 and lv\_line do

1t\_Scarr2.carrid[ :lv\_index ] = :1t\_Scarr1.carrid[ :lv\_index ]; lt Scarr2.carrname[ :lv\_index 1 = 1t Scarri.carrname [ :lv\_index ];

1t Scarr2.currcode :lv\_index ] =:1t Scarr1.currcode:lv\_index ];

lv\_index = lv\_index + 1;

END WHILE

END IF :

et scarr= SELECT from scarr

ENDMETHOD.

WHERE mandt Session\_context( 'CLIENT' );

ENDCLASS.

\*\*\*\* Operation 2 \*\*\*\*

CLASS zelamdp\_explore DEFINITION PUBLIC FINAL CREATE PUBLIC. PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

TRY IT FI

This site is managed by a

Prasad. You want to share y via email: info@sapcodes.com

CLASS-METHODS: get\_info EXPORTING VALUE (et scarr) TYPE ty\_scarr

RAISING Cx\_amdp\_error.

ENDCLASS.

CLASS zel amdp\_explore IMPLEMENTATION.

METHOD get info BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT OPTIONS READ-ONLY USING scarr.

DECLARE 1t Scarri TABLE (carrid "SABAP.type( s carr\_id )",

carrname "SABAP.type(s\_carrname )")

DECLARE 1t Scarr2 TABLE (carrid "SABAP.type( s\_carr\_id )", 4 carrname "SABAP.type(s\_carrname )" );

affiliated with SAP SE

DECLARE 1v counter INTEGER = 1;

DECLARE lv carrid "SABAP.type( s carr\_id )";

WHILE :1v counter BETWEEN 1 AND 3 DO

SAP-Code

Follow Pa

08

if :lv\_counter 1 THEN

1v carrid = 'AA':

ELSEIF :lv\_counter = 2 THEN

lv\_carrid 'AB';

else

lv\_carrid "AC";

END IF

LT CARR1 CARRIDI 11:1v\_carrid: LT CARRI.CARRNAME [ 1 ] = 'Amerian Airline';

it scarr2 SELECT FROM :1t\_scarr2

UNION

SELECT FROM :lt\_scarri

lv\_counter:lv\_counter + 1;

end WHILE

ENDMETHOD.

ENDCLASS.

\*\*\*\* Operation 3 \*\*\*\*