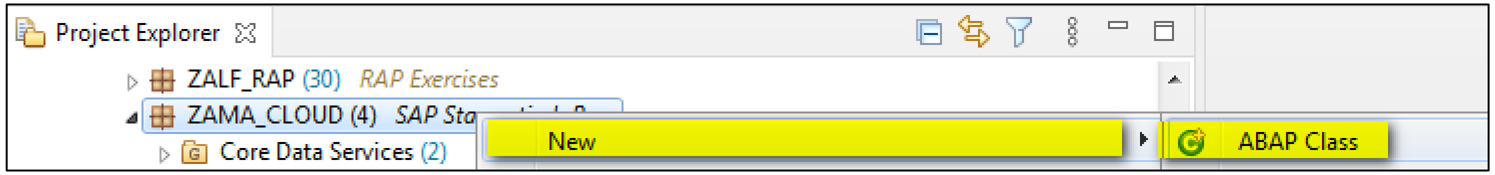
Core Data Services (CDS)

Tutorial 03: Call a Core Data Service from ABAP coding

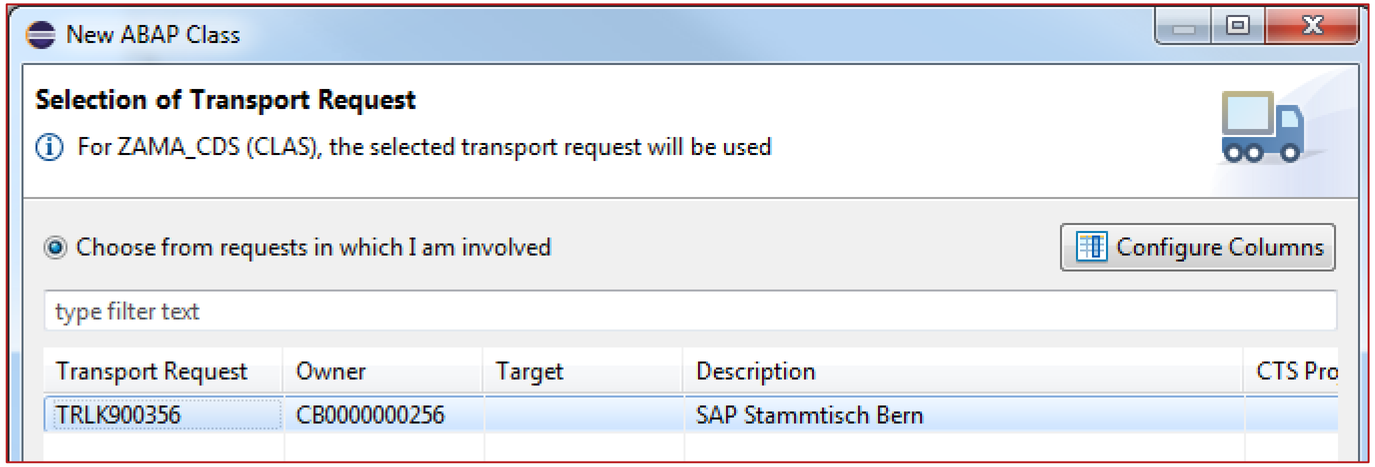
Alexander Maetzing, 13.11.2020

Step 01 - Create a new ABAP class

- Create a new ABAP class following any naming convention you have. Be aware of the uniqueness of the class name!



- Do the assignment to a transport.



Selection of Transport Request

For ZAMA_CDS (CLAS), the selected transport request will be used

● Choose from requests in which I am involved

Transport Request	Owner	Target	Description	CTS Pro
TRLK900356	CB0000000256		SAP Stammtisch Bern	

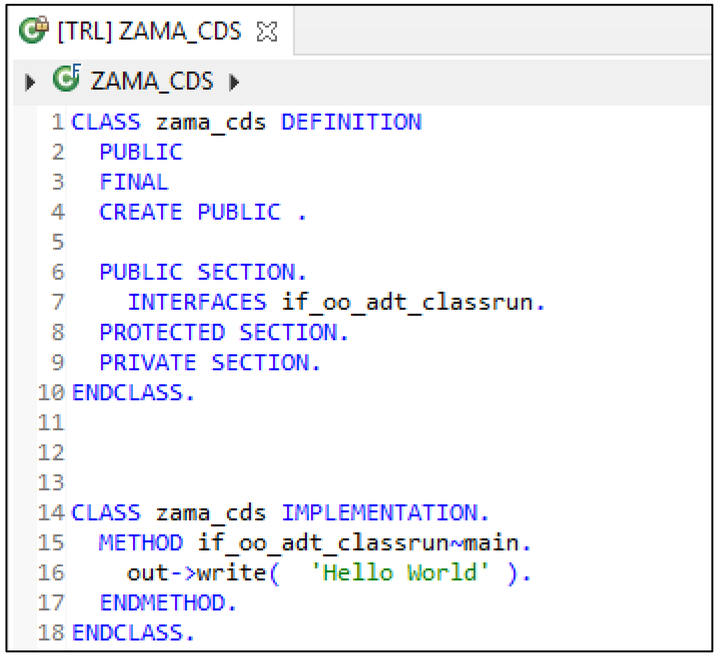
Step 02 - Implement an Interface

- Contrary to the usage of an OnPremise SAP NetWeaver, in the SAP Cloud Platform executable programs or the usual output mechanisms (`WRITE`, ALV, `CL_DEMO_OUTPUT`) are not allowed.

  As a workaround ADT offers the possibility to have an access to the console of the Eclipse development environment. This is done by implementing the Interface `IF_OO_ADT_CLASSRUN`. This interface defines a method `main()` which is called automatically by the ADT framework inside Eclipse.

  The method `main()` has an import parameter `out` (typed to `IF_OO_ADT _CLASSRUN_OUT`), which can be used to interact with the ADT console.

- Implement the method `main()` in that way, that a string "Hello World!" is written into the console.



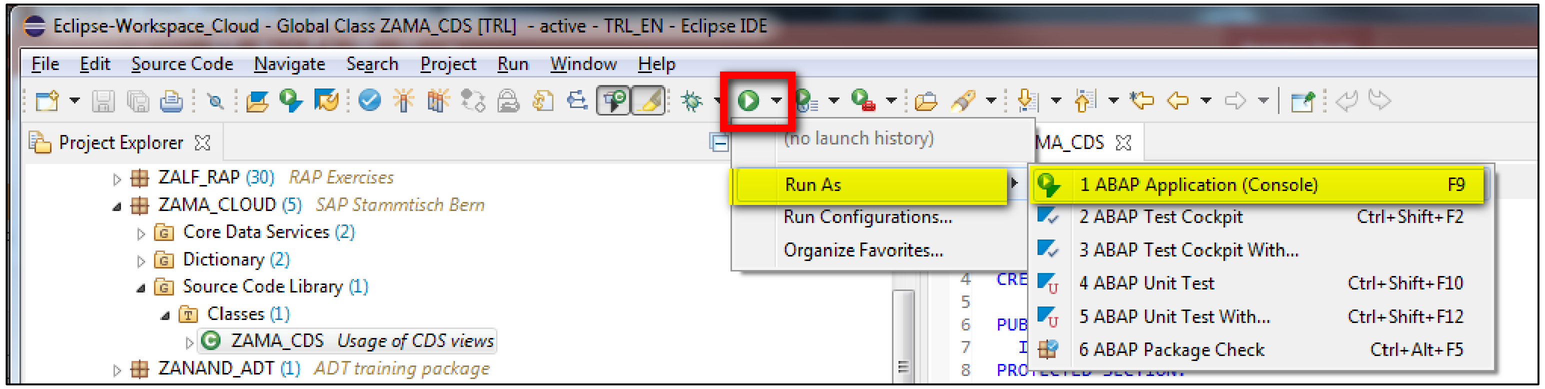
[TRL] ZAMA_CDS

ZAMA_CDS

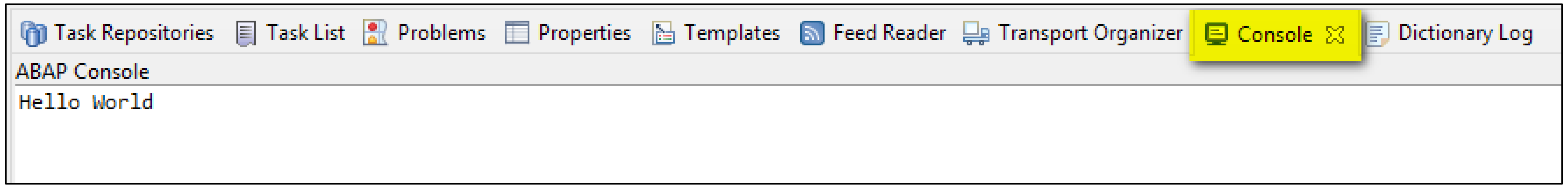
```
CLASS zama_cds DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_oo_adt_classrun.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS zama_cds IMPLEMENTATION.
  METHOD if_oo_adt_classrun~main.
    out->write( 'Hello World' ).
  ENDMETHOD.
ENDCLASS.
```

Step 03 - Process the class

- Process the class as an ABAP program for the console.

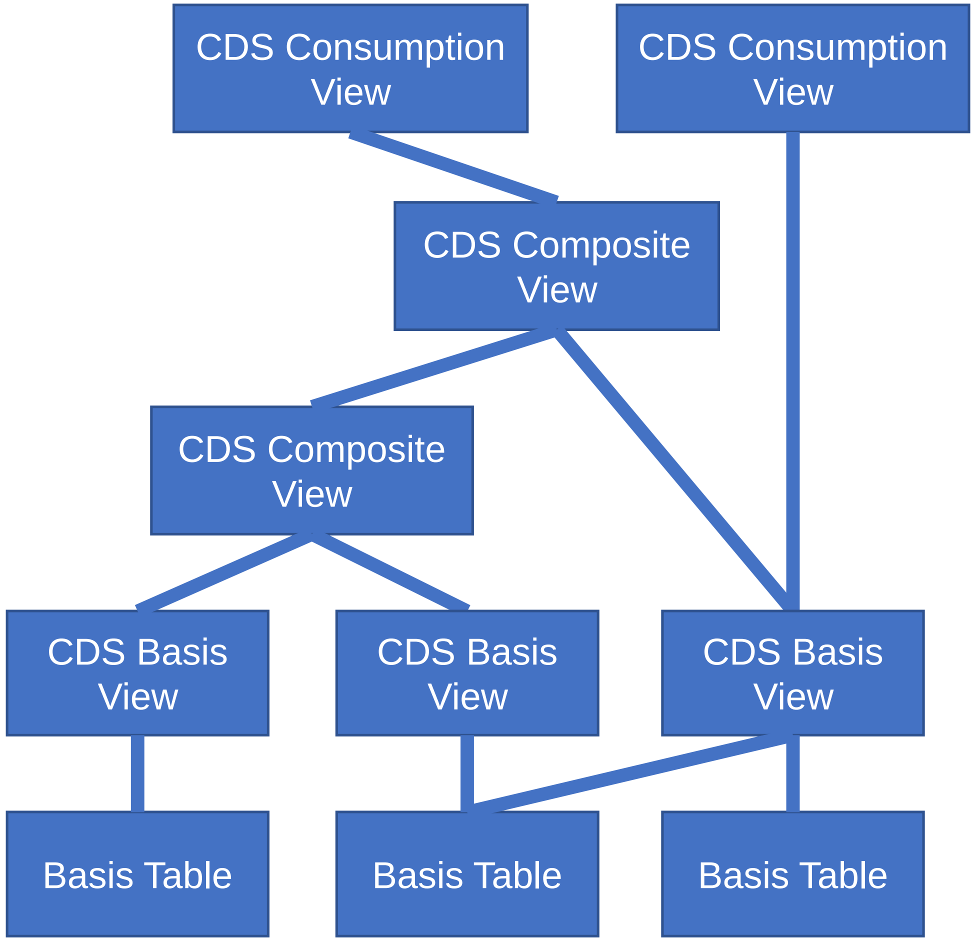

Eclipse-Workspace_Cloud - Global Class ZAMA_CDS [TRL] - active - TRL_EN - Eclipse IDE
File Edit Source Code Navigate Search Project Run Window Help
Project Explorer
ZALF_RAP (30) RAP Exercises
ZAMA_CLOUD (5) SAP Stammtisch Bern
Core Data Services (2)
Dictionary (2)
Source Code Library (1)
Classes (1)
ZAMA_CDS Usage of CDS views
ZANAND_ADT (1) ADT training package
(no launch history)
Run As
Run Configurations...
Organize Favorites...
1 ABAP Application (Console) F9
2 ABAP Test Cockpit Ctrl+Shift+F2
3 ABAP Test Cockpit With...
4 ABAP Unit Test Ctrl+Shift+F10
5 ABAP Unit Test With... Ctrl+Shift+F12
6 ABAP Package Check Ctrl+Alt+F5

- See the console output.



```
Hello World
```

Repetition: CDS Viewtypes of SAP Virtual Data Model (VDM)



CDS Consumption View: Expose the data to the access of different consumers (ABAP reports, analytics tools, ...).

C_<CDS name>, ZC_<CDS name>

CDS Composite (Interface) View: Middle layer to combine, manipulate and process data of different CDS Basic Views.

I_<CDS name>, ZI_<CDS name>

CDS Basic (Interface) View: Fetching the raw data from the real database tables by filtering the access to columns and doing an initial processing.

I_<CDS name>, ZI_<CDS name>

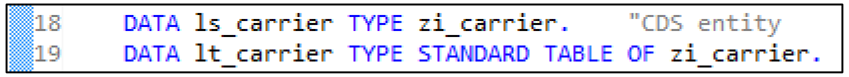
Basis Tables: The database tables containing the raw data.

Naming convention
https://blog.sap-press.com/how-to-name-a-virtual-data-model-in-sap-s4hana

Step 04 - Access the CDS in an ABAP SELECT (1)

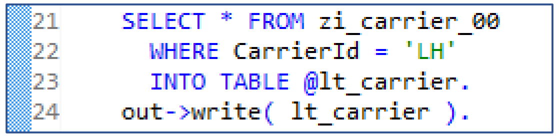
- Using a CDS-View in an ABAP program should - following the naming and usage conventions - happen only with a Consumption View ZC_CARRIER_00 which should be derived from the Basic Interface View ZI_CARRIER_00. Because of simplicity reasons we are using the Basic Interface View as data source directly instead.

- Use the CDS entity in ABAP as a template / type for ABAP variables.



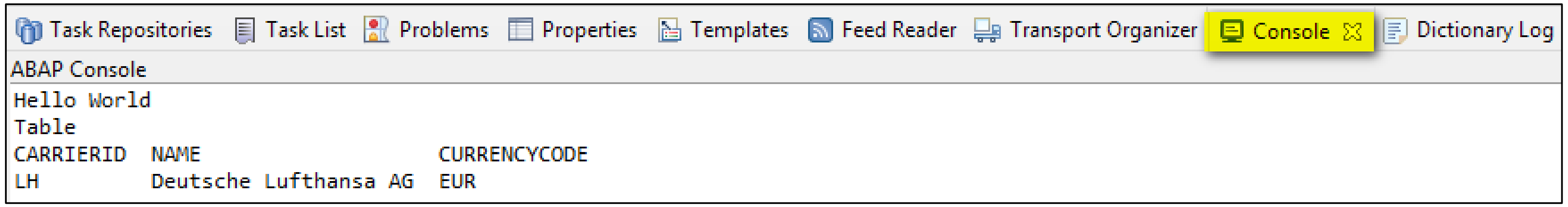
```
    DATA ls_carrier TYPE zi_carrier.     "CDS entity
    DATA lt_carrier TYPE STANDARD TABLE OF zi_carrier.
```

- Use the CDS entity in ABAP as a datasource for an ABAP SELECT statement. Be aware that in the SAP Cloud Platform ABAP SQL instead of OPEN SQL has to be used!



```
    SELECT * FROM zi_carrier_00
      WHERE CarrierId = 'LH'
      INTO TABLE @lt_carrier.
    out->write( lt_carrier ).
```

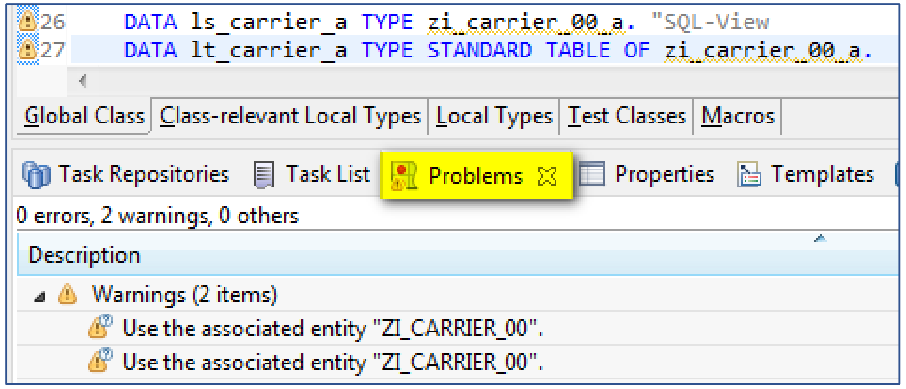
- See the output in the ADT console.



```
Hello World
Table
CARRIERID  NAME                   CURRENCYCODE
LH         Deutsche Lufthansa AG  EUR
```

Step 04 - Access the CDS in an ABAP SELECT (2)

- Use the SQL-View in ABAP as a template for ABAP variables.



```
DATA ls_carrier_a TYPE zi_carrier_00_a. "SQL-View
DATA lt_carrier_a TYPE STANDARD TABLE OF zi_carrier_00_a.
```

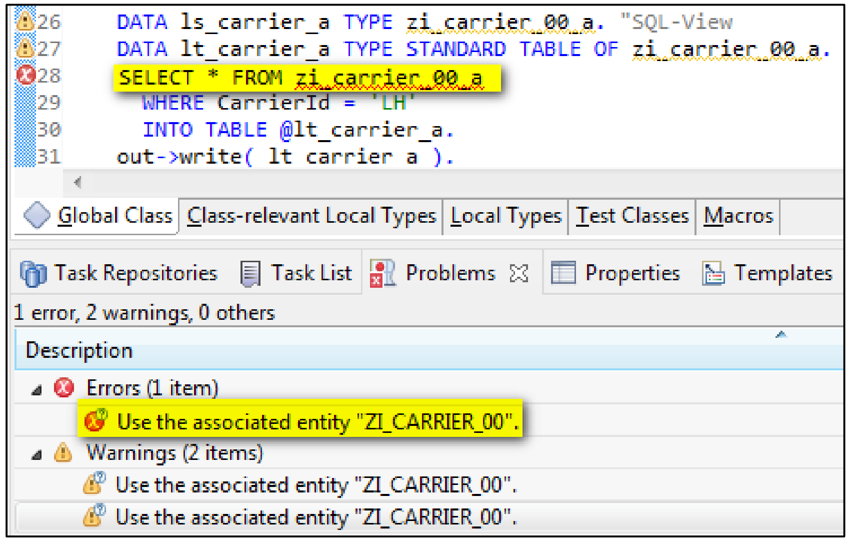
Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

Task Repositories | Task List | Problems | Properties | Templates

0 errors, 2 warnings, 0 others

Description
- Warnings (2 items)
  - Use the associated entity "ZI_CARRIER_00".
  - Use the associated entity "ZI_CARRIER_00".

- Try to use the SQL-View in ABAP as a datasource for an ABAP SELECT statement. Be aware that in the SAP Cloud Platform (contrary to OnPremise SAP NetWeaver) this is forbidden!



```
DATA ls_carrier_a TYPE zi_carrier_00_a. "SQL-View
DATA lt_carrier_a TYPE STANDARD TABLE OF zi_carrier_00_a.
SELECT * FROM zi_carrier_00_a
  WHERE CarrierId = 'LH'
  INTO TABLE @lt_carrier_a.
out->write( lt_carrier_a ).
```

Global Class | Class-relevant Local Types | Local Types | Test Classes | Macros

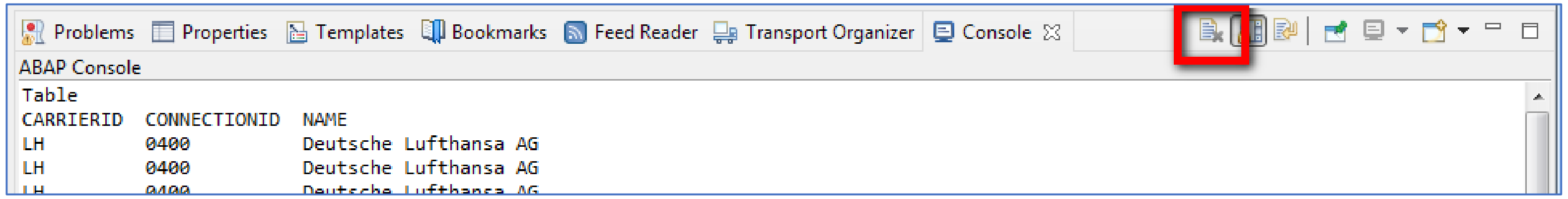
Task Repositories | Task List | Problems | Properties | Templates

1 error, 2 warnings, 0 others

Description
- Errors (1 item)
  - Use the associated entity "ZI_CARRIER_00".
- Warnings (2 items)
  - Use the associated entity "ZI_CARRIER_00".
  - Use the associated entity "ZI_CARRIER_00".

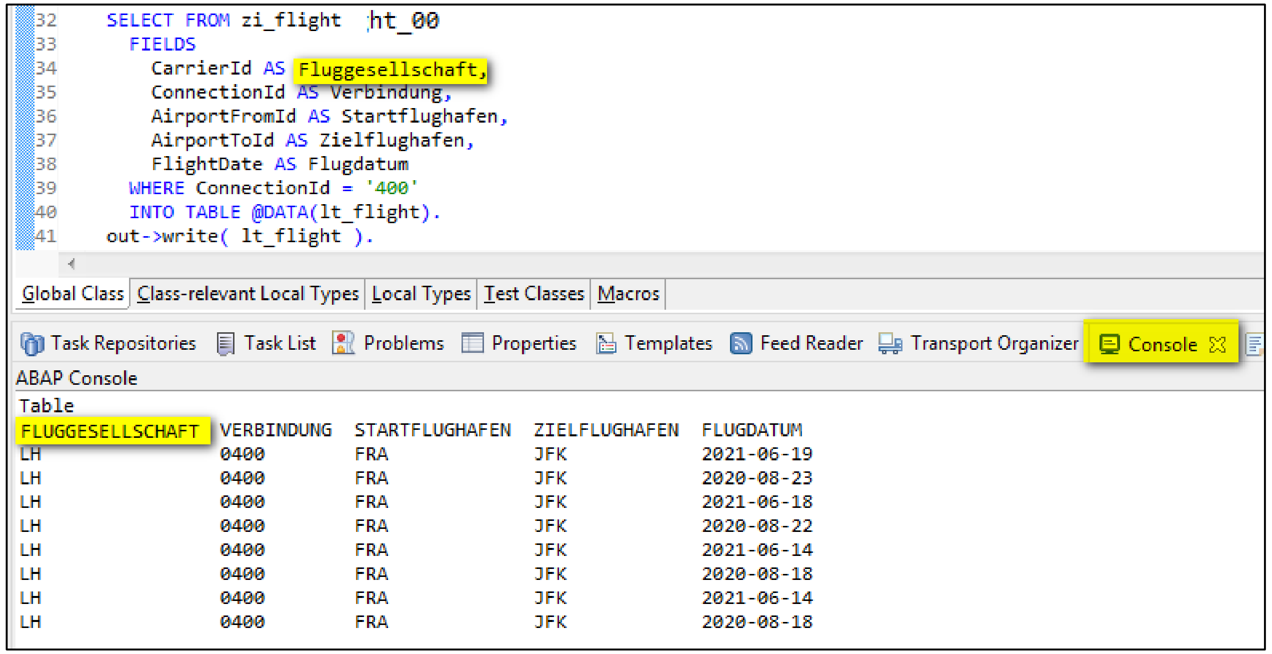
Step 04 - Access the CDS in an ABAP SELECT (3)

- Take care that the output in the console is deleted regularly, to see newly added content at the bottom.



```
ABAP Console
Table
CARRIERID  CONNECTIONID  NAME
LH         0400          Deutsche Lufthansa AG
LH         0400          Deutsche Lufthansa AG
```

- Use the full advantage of ABAP SQL for an access to a CDS.



```
SELECT FROM zi_flight  ht_00
  FIELDS
    CarrierId AS Fluggesellschaft,
    ConnectionId AS Verbindung,
    AirportFromId AS Startflughafen,
    AirportToId AS Zielflughafen,
    FlightDate AS Flugdatum
  WHERE ConnectionId = '400'
  INTO TABLE @DATA(lt_flight).
out->write( lt_flight ).
```

```
ABAP Console
Table
FLUGGESELLSCHAFT  VERBINDUNG  STARTFLUGHAFEN  ZIELFLUGHAFEN  FLUGDATUM
LH                0400        FRA             JFK            2021-06-19
LH                0400        FRA             JFK            2020-08-23
LH                0400        FRA             JFK            2021-06-18
LH                0400        FRA             JFK            2020-08-22
LH                0400        FRA             JFK            2021-06-14
LH                0400        FRA             JFK            2020-08-18
LH                0400        FRA             JFK            2021-06-14
LH                0400        FRA             JFK            2020-08-18
```

Appendix - Sourcecode

```
CLASS zama_cds DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES if_oo_adt_classrun.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
```

```
CLASS zama_cds IMPLEMENTATION.
  METHOD if_oo_adt_classrun~main.
    out->write( 'Hello World' ).

    DATA ls_carrier TYPE zi_carrier.    "CDS entity
    DATA lt_carrier TYPE STANDARD TABLE OF zi_carrier.

    SELECT * FROM zi_carrier_00
      WHERE CarrierId = 'LH'
      INTO TABLE @lt_carrier.
    out->write( lt_carrier ).

    DATA ls_carrier_a TYPE zi_carrier_00_a. "SQL-View
    DATA lt_carrier_a TYPE STANDARD TABLE OF zi_carrier_00_a.
*    SELECT * FROM zi_carrier_00_a
*      WHERE CarrierId = 'LH'
*      INTO TABLE @lt_carrier_a.
    out->write( lt_carrier_a ).

    SELECT FROM zi_flight_00
      FIELDS
        CarrierId AS Fluggesellschaft,
        ConnectionId AS Verbindung,
        AirportFromId AS Startflughafen,
        AirportToId AS Zielflughafen,
        FlightDate AS Flugdatum
      WHERE ConnectionId = '400'
      INTO TABLE @DATA(lt_flight).
    out->write( lt_carrier_a ).

  ENDMETHOD.
ENDCLASS.
```