

cONFIDENTIAL

ABAP RESTful Application Programming Model (RAP)

Hands On: Develop a Read Only List Report

CONTENTS

[Introduction 3](#_Toc163812150)

[1. Create ABAP Package 3](#_Toc163812151)

[2. Create Persistent Database Tables 3](#_Toc163812152)

[3. Create CDS Entities 6](#_Toc163812153)

[4. Enrich CDS entities with UI Annotations 8](#_Toc163812154)

[5. Create a RAP Business Service 11](#_Toc163812155)

[6. Preview your read only application 11](#_Toc163812156)

# Introduction

**Use case:** Travel agents require a read only list of travel requests. From the list the agent should be able to navigate into the associated bookings and view detailed information about each booking of that travel request.

**Note:** Replace \_### with your own unique number to differentiate from objects created by your colleagues in the same system.

# Create ABAP Package

* 1. Via ADT Project Explorer, right click your ABAP system and select *New à ABAP Package.*
  + Name: zrap\_enablement###
  + Description: Package for Travel Application
  + Software Component: choose either:
* ZLOCAL (SAP BTP ABAP and SAP S/4HANA Cloud)
* HOME (SAP S/4HANA)
  1. Create a new transport request and finish.
  2. Right click on *Favorite Packages à Add Package* à Assign your new Package zrap\_enablement###.

# Create Persistent Database Tables

* Create two database tables for the travel and booking data, reusing data elements from installed /dmo content package.
* An ABAP Class will be created and executed once to fill the tables with sample data from the installed /dmo content.
  1. In ADT right click on your ABAP Package and choose *New à Other à Dictionary à Database Table.*
  2. Choose your transport request and finish.
  3. Create the travel table using source code and naming convention provided, then activate.

@EndUserText.label : 'Travel table'

@AbapCatalog.enhancement.category : #NOT\_EXTENSIBLE

@AbapCatalog.tableCategory : #TRANSPARENT

@AbapCatalog.deliveryClass : #A

@AbapCatalog.dataMaintenance : #RESTRICTED

define table ztravel\_a### {

key client : abap.clnt not null;

key travel\_uuid : sysuuid\_x16 not null;

travel\_id : /dmo/travel\_id;

agency\_id : /dmo/agency\_id;

customer\_id : /dmo/customer\_id;

begin\_date : /dmo/begin\_date;

end\_date : /dmo/end\_date;

@Semantics.amount.currencyCode : '/dmo/travel.currency\_code'

booking\_fee : /dmo/booking\_fee;

@Semantics.amount.currencyCode : '/dmo/travel.currency\_code'

total\_price : /dmo/total\_price;

currency\_code : /dmo/currency\_code;

description : /dmo/description;

overall\_status : /dmo/overall\_status;

created\_by : syuname;

created\_at : timestampl;

last\_changed\_by : syuname;

last\_changed\_at : timestampl;

local\_last\_changed\_at : timestampl;

}

* 1. Create a second booking table using the source code and naming convention provided, then activate.

@EndUserText.label : 'Booking table'

@AbapCatalog.enhancement.category : #NOT\_EXTENSIBLE

@AbapCatalog.tableCategory : #TRANSPARENT

@AbapCatalog.deliveryClass : #A

@AbapCatalog.dataMaintenance : #RESTRICTED

define table zbooking\_a### {

key client : mandt not null;

key booking\_uuid : sysuuid\_x16 not null;

travel\_uuid : sysuuid\_x16 not null;

booking\_id : /dmo/booking\_id;

booking\_date : /dmo/booking\_date;

customer\_id : /dmo/customer\_id;

carrier\_id : /dmo/carrier\_id;

connection\_id : /dmo/connection\_id;

flight\_date : /dmo/flight\_date;

@Semantics.amount.currencyCode : '/dmo/booking.currency\_code'

flight\_price : /dmo/flight\_price;

currency\_code : /dmo/currency\_code;

created\_by : syuname;

last\_changed\_by : syuname;

local\_last\_changed\_at : timestampl;

}

* 1. In ADT right click on your ABAP Package and choose *New -> ABAP Class.*
  2. Implement the ABAP Class using the source code and naming convention provided, then activate.
  + Name: zcl\_table\_data\_generator###
  + Description: Sample data generator

CLASS zcl\_table\_data\_generator### DEFINITION

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES if\_oo\_adt\_classrun.

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS zcl\_table\_data\_generator### IMPLEMENTATION.

METHOD if\_oo\_adt\_classrun~main.

DELETE FROM ztravel\_a###.

DELETE FROM zbooking\_a###.

" insert travel demo data

INSERT ztravel\_a### FROM (

SELECT

FROM /dmo/travel

FIELDS

uuid( ) AS travel\_uuid ,

travel\_id AS travel\_id ,

agency\_id AS agency\_id ,

customer\_id AS customer\_id ,

begin\_date AS begin\_date ,

end\_date AS end\_date ,

booking\_fee AS booking\_fee ,

total\_price AS total\_price ,

currency\_code AS currency\_code ,

description AS description ,

CASE status

WHEN 'B' THEN 'A' " accepted

WHEN 'X' THEN 'X' " cancelled

ELSE 'O' " open

END AS overall\_status ,

createdby AS created\_by ,

createdat AS created\_at ,

lastchangedby AS last\_changed\_by ,

lastchangedat AS last\_changed\_at ,

lastchangedat AS local\_last\_changed\_at

ORDER BY travel\_id UP TO 200 ROWS

).

COMMIT WORK.

" insert booking demo data

INSERT zbooking\_a### FROM (

SELECT

FROM /dmo/booking AS booking

JOIN ztravel\_a### AS z

ON booking~travel\_id = z~travel\_id

FIELDS

uuid( ) AS booking\_uuid ,

z~travel\_uuid AS travel\_uuid ,

booking~booking\_id AS booking\_id ,

booking~booking\_date AS booking\_date ,

booking~customer\_id AS customer\_id ,

booking~carrier\_id AS carrier\_id ,

booking~connection\_id AS connection\_id ,

booking~flight\_date AS flight\_date ,

booking~flight\_price AS flight\_price ,

booking~currency\_code AS currency\_code ,

z~created\_by AS created\_by ,

z~last\_changed\_by AS last\_changed\_by ,

z~last\_changed\_at AS local\_last\_changed\_by

).

COMMIT WORK.

out->write( 'Travel and booking demo data inserted.' ).

ENDMETHOD.

ENDCLASS.

* 1. Right click activated class and *Run As -> ABAP Application (Console).*
  2. Check test data in your travel and booking tables by right clicking and choosing *Open With à Data Preview.*

# Create CDS Entities

* Create two base CDS entities to model the travel table as header (root entity) and booking table as item (composition entity).
* The CDS entities will reuse CDS views from the installed /dmo content package to provide additional data for customers, agencies, currencies, carriers, connections, and flights.
  1. Right click on the travel table created and choose *New Data Definition*, use source code and naming convention provided.
  2. Tip: Template “Define root view entity” helps you to format your CDS definition.
  3. **Do not activate yet.**

@EndUserText.label: 'Travel base CDS'

@AccessControl.authorizationCheck: #NOT\_REQUIRED

@Metadata.allowExtensions: true

define root view entity zr\_travel### as select from ztravel\_a### as Travel

composition [0..\*] of zr\_booking### as \_Booking

association [0..1] to /DMO/I\_Agency as \_Agency on $projection.AgencyID = \_Agency.AgencyID

association [0..1] to /DMO/I\_Customer as \_Customer on $projection.CustomerID = \_Customer.CustomerID

association [0..1] to I\_Currency as \_Currency on $projection.CurrencyCode = \_Currency.Currency

{

key travel\_uuid as TravelUUID,

travel\_id as TravelID,

agency\_id as AgencyID,

customer\_id as CustomerID,

begin\_date as BeginDate,

end\_date as EndDate,

@Semantics.amount.currencyCode: 'CurrencyCode'

booking\_fee as BookingFee,

@Semantics.amount.currencyCode: 'CurrencyCode'

total\_price as TotalPrice,

currency\_code as CurrencyCode,

description as Description,

overall\_status as TravelStatus,

@Semantics.user.createdBy: true

created\_by as CreatedBy,

@Semantics.systemDateTime.createdAt: true

created\_at as CreatedAt,

@Semantics.user.lastChangedBy: true

last\_changed\_by as LastChangedBy,

@Semantics.systemDateTime.lastChangedAt: true

last\_changed\_at as LastChangedAt,

@Semantics.systemDateTime.localInstanceLastChangedAt: true

local\_last\_changed\_at as LocalLastChangedAt,

/\* associations \*/

\_Booking,

\_Agency,

\_Customer,

\_Currency

}

* 1. Right click on the booking table created and choose *New Data Definition*, use source code and naming convention provided.
  2. Tip: Template “Define View Entity With to-parent Association” helps you to format your CDS definition.
  3. **Do not activate yet.**

@EndUserText.label: 'Booking base CDS'

@AccessControl.authorizationCheck: #NOT\_REQUIRED

@Metadata.allowExtensions: true

define view entity zr\_booking### as select from zbooking\_a### as Booking

association to parent zr\_travel### as \_Travel on $projection.TravelUUID = \_Travel.TravelUUID

association [1..1] to /DMO/I\_Customer as \_Customer on $projection.CustomerID = \_Customer.CustomerID

association [1..1] to /DMO/I\_Carrier as \_Carrier on $projection.CarrierID = \_Carrier.AirlineID

association [1..1] to /DMO/I\_Connection as \_Connection on $projection.CarrierID = \_Connection.AirlineID

and $projection.ConnectionID = \_Connection.ConnectionID

association [1..1] to /DMO/I\_Flight as \_Flight on $projection.CarrierID = \_Flight.AirlineID

and $projection.ConnectionID = \_Flight.ConnectionID

and $projection.FlightDate = \_Flight.FlightDate

association [0..1] to I\_Currency as \_Currency on $projection.CurrencyCode = \_Currency.Currency

{

key booking\_uuid as BookingUUID,

travel\_uuid as TravelUUID,

booking\_id as BookingID,

booking\_date as BookingDate,

customer\_id as CustomerID,

carrier\_id as CarrierID,

connection\_id as ConnectionID,

flight\_date as FlightDate,

@Semantics.amount.currencyCode: 'CurrencyCode'

flight\_price as FlightPrice,

currency\_code as CurrencyCode,

@Semantics.user.createdBy: true

created\_by as CreatedBy,

@Semantics.user.lastChangedBy: true

last\_changed\_by as LastChangedBy,

/\* associations \*/

\_Travel,

\_Customer,

\_Carrier,

\_Connection,

\_Flight,

\_Currency

}

* 1. Activate both CDS entities together.
  2. Check data from your travel and booking CDS entities by right clicking and choosing *Open With à Data Preview.*

# Enrich CDS entities with UI Annotations

Create a metadata extension and annotate the CDS root and child entity with UI specific annotations.

|  |  |
| --- | --- |
| UI.headerInfo | Headline of list report |
| UI.presentationVariant | Sort order of list report |
| UI.lineItem | Columns displayed in initial list report |
| UI.facet | Object page UI sections |
| UI.facet.type #IDENTIFICATION\_REFERENCE | Fields annotated with UI.Identification are listed in the facet |
| UI.facet.type #LINEITEM\_REFERENCE | Fields annotated with UI.lineItem are listed in the facet as a table |
| UI.facet\_targetElement | Fields in facet are served from target association |
| UI.selectionField | Fields annotated with UI.selectionField are added to the filter bar on initial list report |

* 1. Right click on CDS root entity zr\_travel### and choose *New Metadata Extension.*
  2. Use naming conventionzc\_travel### and the source code provided, then activate.

@Metadata.layer: #CORE

@UI: {

headerInfo: { typeName: 'Travel',

typeNamePlural: 'Travels',

title: { type: #STANDARD, label: 'Travel', value: 'TravelID' } },

presentationVariant: [{ sortOrder: [{ by: 'TravelID', direction: #DESC }] }] }

// annotate the data to be displayed in the facets

annotate entity zr\_travel###

with

{

@UI.facet: [ { id: 'Travel',

purpose: #STANDARD,

type: #IDENTIFICATION\_REFERENCE,

label: 'Travel',

position: 10 },

{ id: 'TravelBookings',

purpose: #STANDARD,

type: #LINEITEM\_REFERENCE,

label: 'Travel Bookings',

position: 20,

targetElement: '\_Booking'} ]

@UI:{ identification: [{ position: 1, label: 'Travel UUID' }] }

TravelUUID;

@UI: { lineItem: [ { position: 20 } ],

identification: [ { position: 20 } ],

selectionField: [ { position: 10 } ] }

TravelID;

@UI: { lineItem: [ { position: 30 } ],

identification: [ { position: 20 } ],

selectionField: [ { position: 20 } ] }

AgencyID;

@UI: { lineItem: [ { position: 40 } ],

identification: [ { position: 30 } ],

selectionField: [ { position: 30 } ] }

CustomerID;

@UI: { identification: [ { position: 40 } ] }

BeginDate;

@UI: { identification: [ { position: 50 } ] }

EndDate;

@UI: { lineItem: [ { position: 50 } ],

identification: [ { position: 60 } ] }

BookingFee;

@UI: { lineItem: [ { position: 60 } ],

identification: [ { position: 70 } ] }

TotalPrice;

@UI: { identification: [ { position: 80 } ] }

Description;

@UI: { lineItem: [ { position: 70 } ],

identification: [ { position: 90 } ] }

TravelStatus;

@UI.hidden: true

LastChangedAt;

@UI.hidden: true

LocalLastChangedAt;

}

* 1. Right click on CDS root entity zr\_booking### and choose *New Metadata Extension.*
  2. Use naming conventionzc\_booking### and the source code provided, then activate.

@Metadata.layer: #CORE

@UI: {

headerInfo: { typeName: 'Booking',

typeNamePlural: 'Bookings',

title: { type: #STANDARD, value: 'BookingID' } } }

annotate entity zr\_booking###

with

{

@UI.facet: [ { id: 'Booking',

purpose: #STANDARD,

type: #IDENTIFICATION\_REFERENCE,

label: 'Booking',

position: 10 } ]

@UI: { identification: [ { position: 10, label: 'Booking UUID' } ] }

BookingUUID;

@UI.hidden: true

TravelUUID;

@UI: { lineItem: [ { position: 20 } ],

identification: [ { position: 20 } ] }

BookingID;

@UI: { lineItem: [ { position: 30 } ],

identification: [ { position: 30 } ] }

BookingDate;

@UI: { lineItem: [ { position: 40 } ],

identification: [ { position: 40 } ] }

CustomerID;

@UI: { lineItem: [ { position: 50 } ],

identification: [ { position: 50 } ] }

CarrierID;

@UI: { lineItem: [ { position: 60 } ],

identification: [ { position: 60 } ] }

ConnectionID;

@UI: { lineItem: [ { position: 70 } ],

identification: [ { position: 70 } ] }

FlightDate;

@UI: { lineItem: [ { position: 80 } ],

identification: [ { position: 80 } ] }

FlightPrice;

}

# Create a RAP Business Service

* 1. Right click the base root entity zr\_travel### and choose *New Service Definition*.
  2. List the entities that will be in scope for the new business service using the source code and naming convention provided, then activate.

@EndUserText.label: 'Travel Service Definition'

define service z\_ui\_travel\_m\_### {

expose zr\_travel### as Travel;

expose zr\_booking### as Booking;

}

* 1. Right click on the service definition created and choose *New Service Binding.*
     + Name: z\_ui\_travel\_m\_###\_02
     + Binding type: ‘ODATA V2 – UI’
  2. Activate the new service binding.
  3. Choose *Publish* to publish the local service endpoint.

# Preview your read only application

5.1 Open the newly published service binding, right click on root entity Travel, and choose ‘Open Fiori Elements Preview application’.