First create:

The root view and the child view

Then create behavior definition

Then create the class from the behavior definition

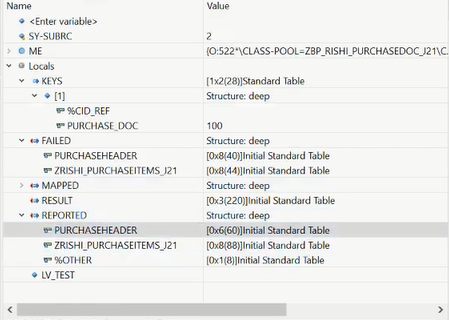
Create new behavior definition on the projection view of header

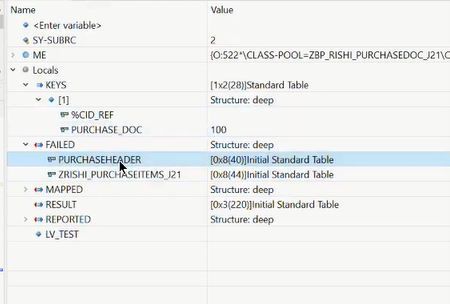
Optimistic locking: 2 or more people can have access to edit the document

Pessimistic locking: if someone is editing then you will not have access to edit the document

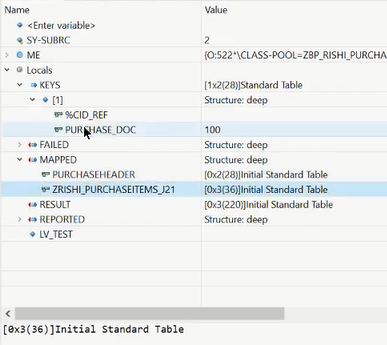
90% of developments happen in eclipse as we cannot create business definitions, services, root views, business objects in BAS.

The only purpose of BAS is to consume the service and see the preview of the app.

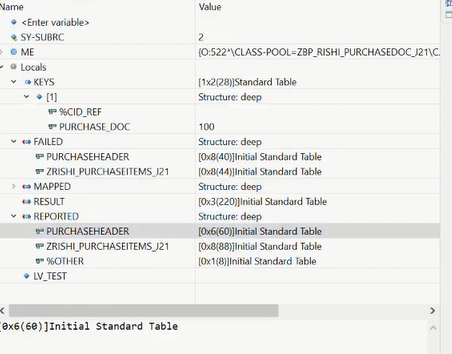




FAILED – will be used to populate any entries which failed or were not processed successfully during create, update delete etc.



MAPPED – will contain the details of the newly create PO’s as in this case



REPORTED – we mention the actual message details such as error, warning success etc.

RESULT – it can be either there or not. So it is not that important.

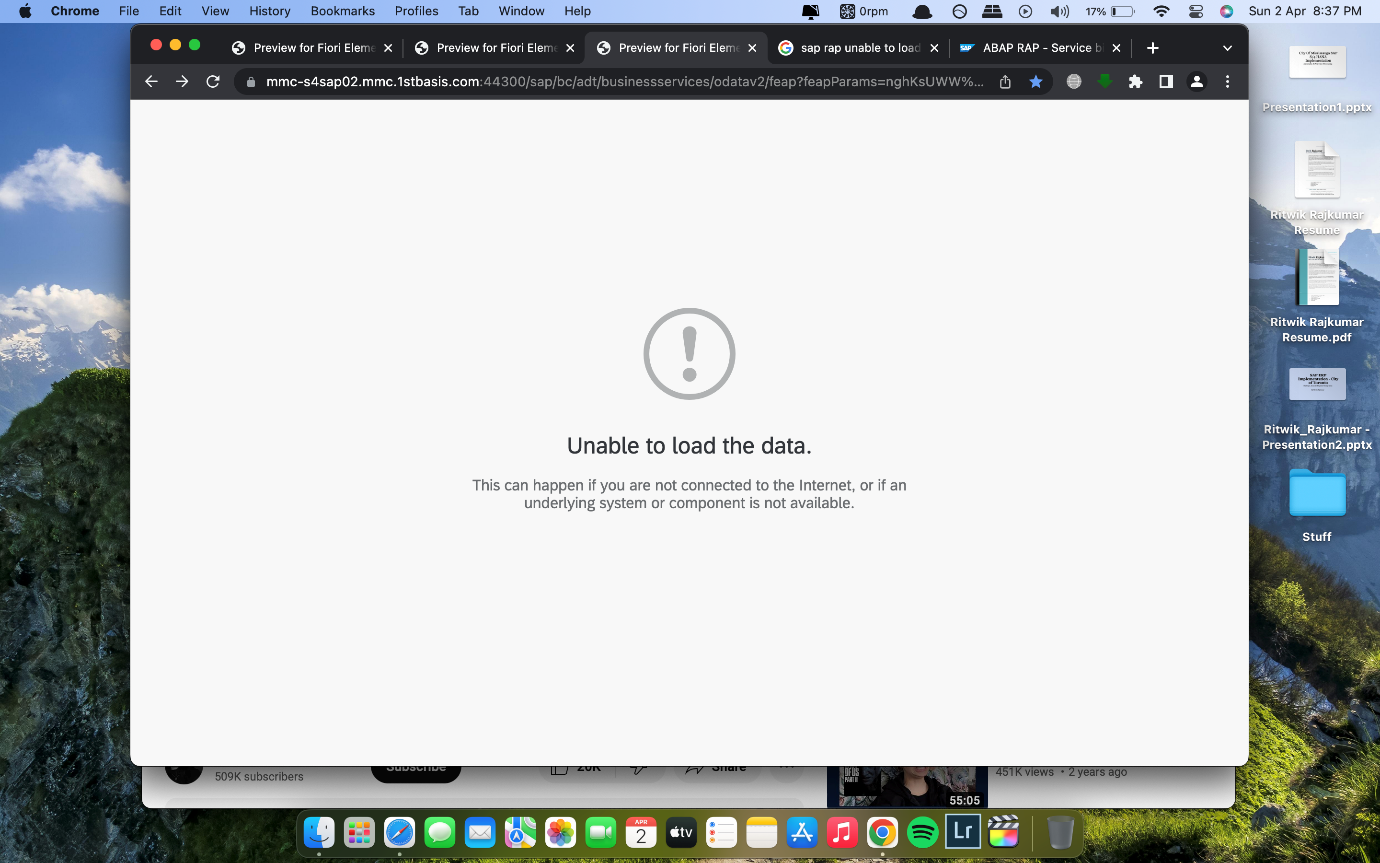
In the behaviour definition we can select whether it is manage/unmanaged scenario.

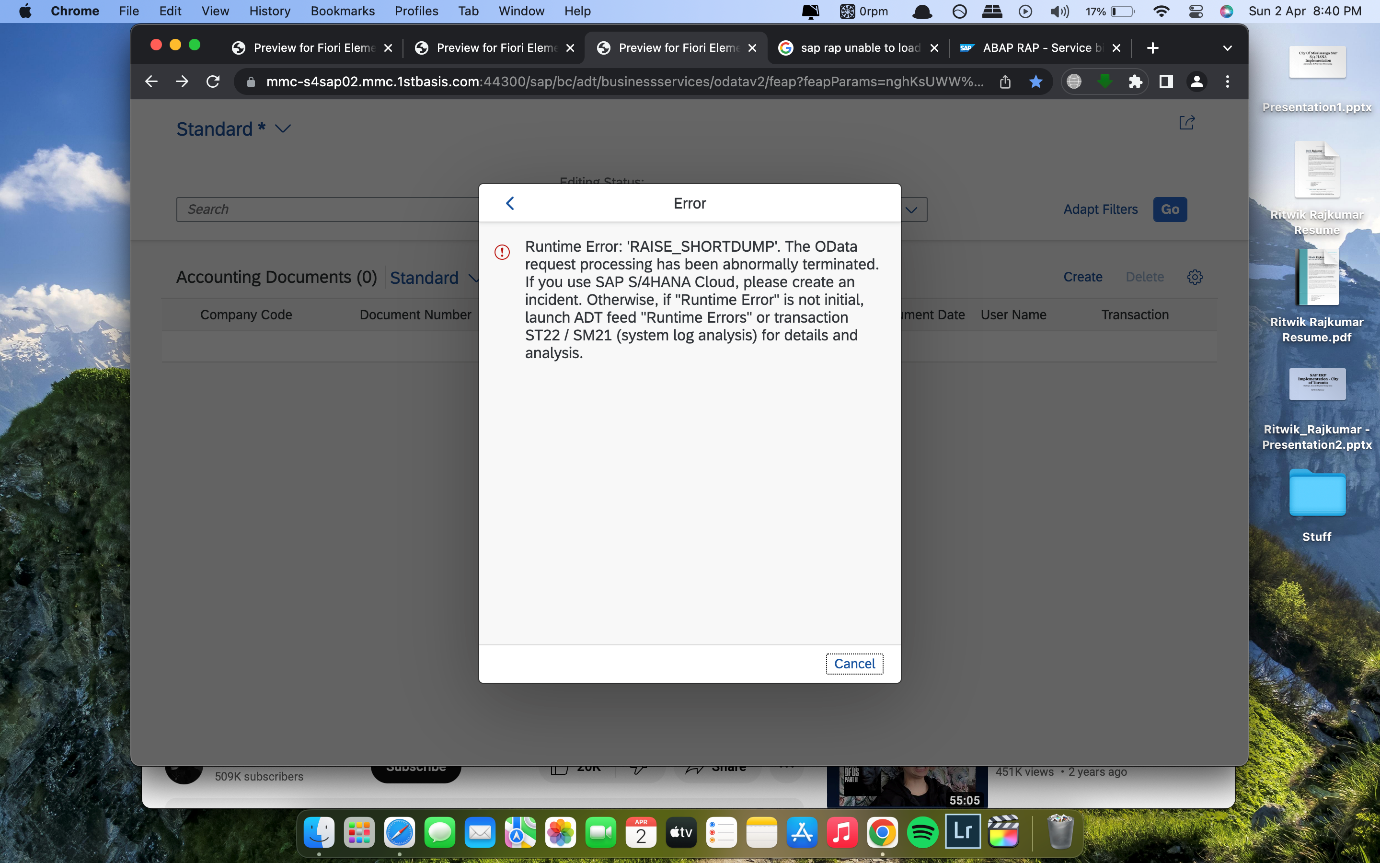
Behaviour definition is created on top of the root view

Behaviour projection is created on top of the consumption view of the root.

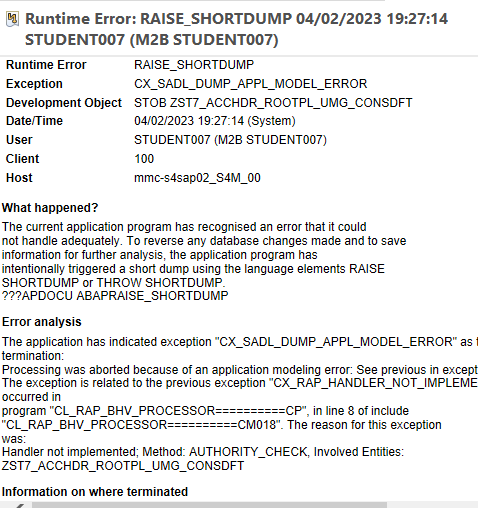
\*Always create the implementation class of the behaviour definition at the end on the basic root view for draft functionality. Especially after the behaviour definition & projection have been done\*

Often you may get an error message when trying to load a page like the following. Like when you click on the Go or Create button:





In Eclipse the error message will be like:



This is because you need to define and then implement an empty method for authorization in the class

Definition:

METHODS authority\_check FOR instance AUTHORIZATION

IMPORTING keys REQUEST requested\_autohrizations FOR AccountingHeader RESULT result.

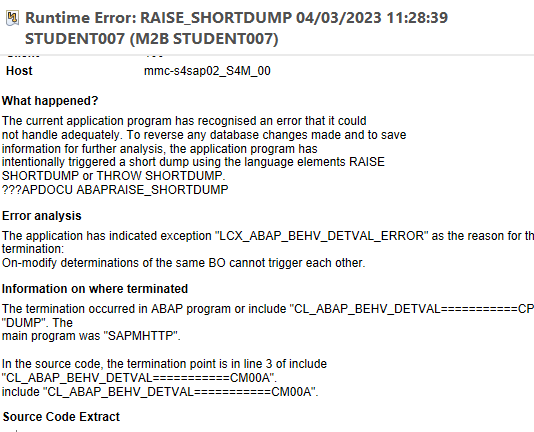
Implementation:

METHOD authority\_check.

ENDMETHOD.

Service definition & service binding are created on top of the root consumption view

In Unmanaged scenario if you get the following error:



Then this could be because in unmanaged scenario

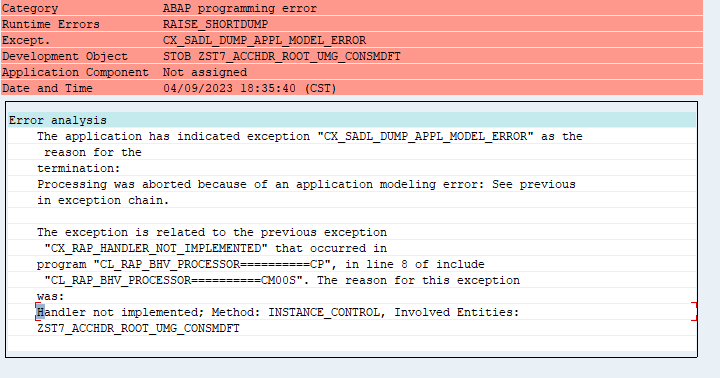
Don’t add the field bukrs value as shown below:

**determination** generateonload **on** **modify** **{** **create;** **field** bukrs**,** belnr**;** **}**

**Use just create:**

**determination** generateonload **on** **modify** **{** **create;**

However, adding fields will work in managed scenario:



If the dump says Handler not implemented; Method: INSTANCE\_CONTROL then check the spelling for the ID and TARGETQUALIFIER in the UI.Facet: in the metadata extension file.

To save data in unmanaged scenario after the user enters the fields on the screen:

There are 3 scenarios:

1. Interaction phase
2. Transactional buffer

First create a buffer class as follows:

1. CLASS lhc\_buffer DEFINITION.
2. \* Defining static variables
3. PUBLIC SECTION.
4. CLASS-DATA: mt\_acc\_hdr TYPE TABLE OF zst7acchdr\_dftum,

mt\_acc\_itm TYPE TABLE OF zst7accitm\_dftum.

1. ENDCLASS.

3. Save sequence

For this we have to use a custom FM, BAPI or Class Method.

Etag: it is used for implementing optimistic locking. Etag depends on the last changed value. So it will have a lastchangeddatetime field assigned to it.

For example: **etag** **master** lastchngdttm

Optimistic locking – at a time two users can edit but only the latest changes are considered.

Real example:

User1: accessed the document at 7.30 AM. Then etag = 7.30 AM

He then starts editing the document and makes some changes, like changes the date

He then saves at 8.30 AM.

Then the lastchngdttm value will be 8.30 AM.

User2: accessed the document at 7.30 AM. Then etag = 7.30 AM

He then starts editing the document and makes some changes like changes the currency code.

But he saves the changes at 8.35 AM

So how will etag work here:

etag = last changed value

So if the etag NE last changed value

Then we will get an error message like ‘The changes have already happened and these changes will be ignored’

For the above logic to work we have to read the last changed value, for this we need to implement the READ method and also the UPDATE method.

Pessimistic locking – at a time only one user can edit

In BAS there are 3 commands to be used to deploy our applications:

1. npm install
2. npm run build (to build the project)
3. abap deploy

on doing abap deploy BAS will ask for the following input from you:

1. Select a SAP target system: enter the virtual name of the system
2. Enter the application name.
3. Enter the description for the application.
4. Enter the package for the deployed application.
5. Enter the transport for the deployed application.

Once the above steps are successfully completed it will create a BSP application which can be checked inside the package in SE80.

Once we get it we then just need to give the name of the BSP application to the BASIS team who will deploy it to the FIORI Launchpad (including creating the catalogue etc).

Managed & Unmanaged queries:

Managed – runtime of the query is taken care by SADL (Service adaption description language). The application developer does not have to have to deal with the construction of the SQL statement to retrieve data from a database table.

SADL – it is an ABAP technology that allows for consumption of an entity relationship like data models in ABAP like model-driven approach.

In the context of SAP HANA, SADL enables fast read access of the database data for scenarios on mobile and desktop applications using query push down.

Unmanaged – if data must e exposed as an ODATA service that can’t be read directly from a CDS view and its underlying database table, we go for an unmanaged query.

The abap query is linked to a class. This class needs to implement an interface IF\_RAP\_QUERY\_PROVIDER and implement whatever data selection is required. For instance, using some legacy ABAP code which cannot be transformed into a CDS data model or fetching data from a different data source other that database tables in the current database server.

AMDP (ABAP Managed Database Procedures):

It allows you to create database procedures on HANA database from the ECC system.

1. When creating AMDP class we need to implement the interface if\_amdp\_marker\_hdb. It marks the class as an AMDP class.
2. AMDP method should only have passed by value parameters.
3. AMDP will work only on HANA databases.

Returning parameters cannot be used in AMDP class.

They have to be passed by value only.

AMDP classes cannot be edited in the SAP backend, they can only be edited in eclipse.

Both normal methods and AMDP methods can co-exist in a class.

There is a standard class that you can refer to CL\_CS\_BOM\_AMDP.

AMDP methods cannot be edited from SAPGUI.

Statements in AMDP are terminated by ;

Differences between AMDP and CDS:

1. AMDP is database dependent while CDS is database independent.
2. AMDP reads and writes while CDS only reads.
3. AMDP we have to use native HANA SQL script only while in CDS we use ABAP open SQL.

AMDP functions do not have exporting or changing parameters. They only have returning parameters. They have only one tabular returning parameter.

We cannot call AMDP function in an ABAP code directly unlike AMDP procedure.

AMDP procedures started supporting AMDP functions from release 7.50.

AMDP procedures do not have returning parameters.

SAP BOPF (Business object processing framework):

It’s mostly applicable for transportation management or SAP EHSM (environment, health safety management) projects.

With ABAP RAP BOPF may not be needed in SAP S4 HANA projects but they are still needed in the above modules.

In BOPF we are going to create business objects (create, update & delete) operations.

BOPF is a step by step procedure to create our BO and manage our BO for transactional purposes.

Service oriented architecture.

Advantages of BOPF:

Rapid prototyping

Reusability

Less implementation effort

More stable

Service & transaction manager API’s:

BO layer: where the BO layer linkages will reside. Then the BO create, update, query will be taken care of.

To access any components of the BO or even CRUD operation in the BO – we have to instantiate the service manager API first-hand.

To perform save operation we have to access or instantiate transaction manager API on first-hand.