

A computer screen shot of a diagram

Description automatically generated

A computer screen shot of a diagram

Description automatically generated

A computer screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a diagram

Description automatically generated

We welcome back.

In this video tutorial, I will discuss with you how we can build SAP Fiori application using RESTful

application programming model.

In the last video tutorial, we have already seen how we can create free Abap trial environment in BTP

and how to connect it with the ADT tool in eclipse.

With that, our development setup is already ready and we can start our programming.

First we will create a database table on top of it.

We will create our CDS view.

On top of that we will create one projection CDs view, then service definition, then service binding.

Then at last we will preview the application.

This flow is there to create a read only SAP Fiori application.

Then we will convert the same application into a transactional application where we can perform, create,

update, delete operations.

For that, we will create create behavior definition it's implementation and behavior projection.

Let's move to edit tool in eclipse to start building our SAP Fiori application.

As you can see here, I have already logged in into Abap cloud project.

Now let's first create a Abap package.

Let's give it a name as Csla demo package.

Okay, so we have successfully created a package.

Now let's add this package in favorite package list.

Now let's create one database table here.

Cute name as the SLA has book.

According to the naming convention followed for RESTful application programming model.

If the database table is a persistent table, then we have to use a underscore in the name of database

table.

If it is a draft table, then we can use d underscore.

We are going to create here a persistent database table.

So that's why we have inserted a underscore here in the name of database table.

Now here to define a database table, we need to provide the field name and the data type.

To save your time I have already written a code.

Now let's activate the database table.

At this moment, we only have created the database table, but there is no data inside it.

As you can see here.

So to create a data into this database table, again I have written a code already.

So I will be using that to populate data into this table.

So for that I need to create.

It Abap class.

Let's give it name as z SQL data, create SLA a book and let's queue description as data creator for

database table.

Csla as book.

As I mentioned earlier, I have already written a code to save your time, so I will be using that here,

right?

Now let's execute it.

Glitter has been created in this database table.

As you can see here, data is successfully created in the database table.

You might be thinking why we are using the class and method here, and why we have not written this

code in the Abap report.

The reason is classical Abap report cannot be created in Abap environment in BTP.

That is why we have used the class and method here to populate data in the database table.

So with this tape we have successfully created a database table and data inside it.

So this step is completed.

Now let's create one CDs view on top of this database table.

To create a CDS view.

Right click on a database table and click New Data Definition.

Let's give it a name as GSL, I as book.

Now let's give a description as CDs.

Buford table DSL a s book.

According to the naming convention, the CDS view name should contain I underscore for interface view.

That's why in the name we have used, I underscore.

Here we need to select defined root view entity.

So as a result, the Indosphere has been created to this CDS view.

We do not have any child, so we do not require this statement here.

And this statement as well.

Napier.

Let's give a proper name to the fields of database table.

Will rename all the fields here.

Now let's activate this series view.

So with this step we have successfully created CDS base data model.

Now let's create data model projection.

Let's give it a name as the SLC s book and description as Rhodesian view of CDs view z sla I s book.

Here you have to select template to define projection view.

Okay so here we got one error.

The root keyword is missing.

Okay, so now that error has gone.

There is a warning is asking to use conjunction of provider contract.

So let's use it.

Okay, fine.

Here.

We do not want to use all the fields that has been projected from underlying interface CDs view.

So I will remove some fields here.

So we will be taking only these fields from the underlying CDS view.

Now let's activate it.

So with this step we have completed the creation of data model projection.

In the next step we will create service definition.

Now let's create service definition.

Right click on the projection view and select option New service definition.

Let's give it a name as the SLI UI is book.

In general, for service definition, there is no need for a prefix or suffix to differentiate meanings.

However, in a use case where both use of same service definition is planned for UI and API services,

the prefix may follow the rules of the service binding.

In our case, we will be creating the service definition for SAP Fiori application.

So that's why we will use the name as GSA underscore s book.

Let's give it the description as service definition for flight booking.

Okay, now let's activate it.

Now with this tape we have completed creation of service definition.

Now let's create service binding.

In case of service binding, use prefix UI underscore.

If this service is exposed as UI service API underscore.

If this service is exposed as web API, use the suffix or two if this service is bound to OData protocol.

Version two.

Underscore zero four if this service is bound to OData version four.

So let's go and create service binding.

So right click on Service Definition and create new service binding.

Now let's give it name as CSL UI as book oh two.

And let's give description as service binding.

Here you need to select binding type.

So since we are going to use it to create a Fiori UI application using OData v2 version, so I'll be

selecting that option.

Okay, let's activate the service binding.

Now with this step, we have successfully created service binding.

Now you have to publish the OData service.

So click here.

This will take a moment to get published.

So as you can see here, service has been published.

So click here and hit preview.

It will take you to the browser if you have already logged in.

No need to log in again and your application will get logged.

So you can see here our Fiori application is lodged.

But as you can see here we are getting message.

There is no visible columns in the table right now.

That means our Fury application got created, but there is no columns visible there.

That is because when we have created projection view, we also need to provide annotations here so that

these fields get displayed on Fury application.

Now let's add annotations here and see the result in the UI.

Let's activate it.

I will explain you all these annotations in a minute.

Let's first look at the result.

Reload this page.

Okay, we got the fields.

Click on go and you will see the data in that table.

So as you can see here the result of adding UI annotations to the projection view.

You can see the fields here along with the table content.

Now let's step back and try to understand the significance of the annotation that we have used.

Your annotation line item, make this field visible on your position.

Specifies the sequence in which these fields will get arranged in UI.

As you can see here.

Position ten.

Position 20.

Position 30.

This can be 1 to 3 as well.

Right here you can give incremental numbers.

Then we have label.

Using these annotation you are simply providing name UI name to the field.

You can see here, right?

You got text as an ID because you have specified label as airline ID then we have these annotation importance

value set to high.

The fields with these annotations will be displayed on screen even if you resize the UI screen.

Let me resize this screen.

I have researched it, but still you can see airline ID, connection ID and passenger name.

Why?

Because those fields are set with the high importance.

Other fields are with low importance.

Passenger name is with high importance.

That's why you could see passenger name, connection ID and airlines here even if I resize it.

Reduce the size of Us3.

Elaine Field.

Passenger name cancellation status.

All these three fields.

Master selection field.

If you go to UI you can see those are here.

So basically using these annotation selection field you can select the value for that field.

Okay.

So basically it helps you to filter the content of table that is being displayed on UI based on these

three filter criteria.

So basically here this selection field is used to filter out the content displayed on UI screen.

Now let's click on this record here.

And you can see the flight booking details.

So this is the object page which being displayed because of the use of this annotation.

Over here you can see the label flight booking details here.

Here we have use a one more annotation called identification.

And then we have provided a label to these annotation.

So these get displayed on object page.

As you can see here statistics for cancelled has space for book.

That is what we are displaying through this label for the annotation identification.

So I hope you understood the importance of using UI annotation at Projection View.

So using UI annotations we are instructing the framework to construct a Fury application.

You can remove these annotations from the projection view and keep it in a separate file for that purpose.

What you can do is simply right click.

Go to Source code and click Extract Metadata Extension and give the name to this file.

It will be same as your projection view name.

Metadata extension for CDS view is the SLC js book.

So now you can see.

This metadata file is created, and if you go back and check your projection view, you will see all

those annotations has been removed.

So this practice you can use to keep the projection view field information and the annotations information

separate.

To use these extension metadata extension.

First you need to allow it in your projection view.

So you have to use annotation metadata X allow extension through.

Now go to your metadata.

Now this error will get resolved.

And here you need to provide metadata layer.

Press control space and it will give you the options out of which I have selected value for here.

Now let's activate it.

And let us go back to the browser and refresh the application page.

Okay.

Still, it is working.

Even if we have separated your annotations from the projection view.

With this example.

I hope you understood how we can create read only application.

Fury application using RESTful application programming model.

No.

We will extend this same application to enable it to perform operations, create and delete.

For that purpose, we need to create CDS, behavioral definition implementation and then behavioral

projection.

So let's first create CDs behavioral definition.

The name given to the CDS behavioral definition will be same as that of the CDS view root entity name.

So right click on the CDS view here and click New Behavioral Definition.

Here you can select the type of implementation managed or unmanaged.

In case of manage implementation system will take care of create, update and delete transactional operations.

We do not run it to write code for it, but if we choose unmanaged then transactional operations create,

delete and update.

We need to write our own code.

So I will select manage.

Okay.

So you can see here the behavioral definition is generated.

This is only the definition.

We now have to implement it.

But this step we have completed see this behavior definition.

Now we need to implement this behavior.

For that purpose click here create behavior implementation class.

So first we need to activate it.

And then click here.

Then select Create Behavior Implementation class.

So for the operations that we have listed here system will automatically generate code for these transactional

operations.

Since we have selected the implementation type as managed.

Right now.

Let's activate this class as well.

So this class has been generated by framework itself.

So with this step we have completed this behavior implementation.

Now let's create behavior projection.

For that right click on projection View and select Behavior Definition.

So as you can see, behavior definition contains.

Create update delete.

Operation.

For projection, you have a option to choose which operations you want to apply to the Fury application.

Let's say I do not want to go with the create part.

In that case, I can simply comment it out.

Now few in the city's behavioral definition.

We need to provide the mapping.

Mapping for database table Ksla, a book.

Basically this helps to map database table fields.

2D CDs.

View fields.

Okay, now let's activate it.

Go back to the web page and reload the application.

So as you can see here we got this option delete.

Let's click here on this record.

Now you can see here edit option is there.

Let's mark this field as X and click save.

So with this operation what we have done is we have updated the record.

So now you can see here the booking status is marked as x.

And if you click here on the checkbox you can also delete that record.

The create option is not there because in the behavioral projection we have commented that operation.

Right.

That's fair.

So if you uncomment it.

And you reload the application.

You will get the create option as well.

So here you can put some values.

They'll create.

So you can see.

That record has been created here.

So I hope with this example you understood how we can create Transactional Fury application using the

RESTful application programming model.

That's it for now.

Have a nice day ahead.