Audio file

[ABAP Webdynpro Training2 - 27042020.mp4](https://capgemini-my.sharepoint.com/personal/abhay_bhagwanjee-singh_capgemini_com/Documents/Transcribed%20Files/ABAP%20Webdynpro%20Training2%20-%2027042020.mp4)

Transcript

Would be displayed at your view.

So there is a mediator between your model and view which is called controller, so controller.

Is going to communicate between view and model.

So it will take the inputs from the user, get it, get the data, process with the help of model and then display the data in the browser with the.

Help of view.

So that is the role of a controller.

So that's why we said our web dynpro is based on an MVC model.

Now let's see the architecture of web dynpro.

Uh, if anybody has any question, they can put it in a comment box or they can unmute and ask any question.

So let's start with an architecture.

Uh, so there's a is a dotted line which says above.

This is an external visibility of your application, and this is your internal application visibility.

Now, how the browser is going to access your application and how there's.

This is an internal picture of your application.

So first we will see how it looks internally, then we will see how the external world is going to communicate with your internal environment.

So inside your internal visibility you can see in this particular picture there is one window and there are a few views created in the application. So let's say this is view one, these is view 2 and this is a view 3C.

All three views are.

Communicated or are integrated into a one single window.

So a window is holding all these 3 views.

And every view has its view controller. So this is V1 which has its own controller, view 2 has its own controller and view three has its own controller.

So number of views the the total number of vusers equivalent to total number of controllers.

There there will be only one.

There's a one single window which will hold all these 3 views and there's there will be one window controller associated to one window.

Now what are these symbols?

So these are called plugs.

Now what are plugs?

These plugs will help us to navigate from one view to other view.

Uh, so if if I created 3 views and I integrated them in the window.

So how I'm going to navigate between those views? How these system will understand? Now from this view I have to move to from V1 to have to move to view 3, then from view 3 have to move to view 2. So the the navigation between these views are handled by plugs.

Uh, there are two plugs.

One is an inbound plug and the other is an outbound plug.

Their their name itself explains what it means.

Like inbound is to came inside of you and outbound is to go outside of you.

So every view will have an inbound and outbound plug, let's say.

My view one is my first view which will be visible in the browser when I open the application.

Which means I have to define this particular view one as.

Uh, before?

Bellevue because this is the one which will be visible when I will open my browser.

So at the window level I will give an uh.

Setting I will give a, give give a I will assign property to this view that it should be a default view.

So when I will open my browser view one will be visible.

In view one I have created one button which onclick of that button I have to navigate to view 2.

So from my default view now I will I will be moving to view 2 using my by using my.

So as we are exiting from view one which means we are going out of view one, so we will be defining 1 outbound plug at view one and we are entering into view 2 which means we will be defining an inbound plug in view 2.

So every view will have an inbound.

Outbound plug, it depends where from where we have to navigate to which view.

Depending upon our navigation we will be defining the plugs inbound or an outbound.

So what is M here?

M stands for method.

Every controller will have a set of methods.

So which will help to process the logic.

Every controller even window controller, view controller, component, controller has a set set of methods.

Which which are executed in a particular sequence and every method is executed when you run an application in a proper sequential manner.

So when we will be designing our application I will show you each and every method.

So we basically call them as in hook methods.

So there are several methods automatically generated by web dynpro and we will while while designing an application we will understand the purpose.

Of each and every month method we will understand at what case we will be writing a logic in which particular method.

So for now.

Uh, we will move ahead.

This was the basic picture how window and view plays a role and what controller is going to handle and how plugs will help us to navigate between these views.

Once we ready with the window and the layout view layout.

There comes a component controller into picture.

As we said, a component control is a global controller, a master controller which will help us to communicate between view and a model.

So it will based on the user inputs, it will help us to get the data from model and show it into a.

So a component control plays an important role here in the MVC architect.

So even a window controller can access a component controller, and every single view can access a component controller.

So component control will also have a set of methods which where we will be putting our common logic which needs to be accessed by each and every view.

Let's move ahead.

So component, uh, similar to component controller, there's a custom controller.

So so we we usually don't go for creating a custom controller because, uh, component controller itself is capable of.

Handling all the.

Master logic so but if in in any scenario where we don't have to put much load on a component controller, we can go for a custom controller which is similar to component controller.

The properties of custom controller is almost same as the component controller.

As shown in this picture, it is also a mediator between your model and your view, so it will it.

It is just like a.

A copy of component controller and it will play a same role to control logic but control the data between the model and view.

Uh, once we are done with your component, controller windows and layout structure that comes into picture component interface.

Now till here we were only looking at the internal visibility of your application.

There was no external visibility.

To your application 'cause we we were only done with Windows and layout and controllers, but how you gonna interact with the external void?

So here comes our component interface.

A component interface will have an associated component interface view and an interface controller.

This interface via an interface controller will have visibility to the external environment the the external external world of Upton Pro, so.

When we when a web application is created and component interfaces are generated automatically it will create one interface view and one interface controller so.

Thus this interface view an interface controller.

This view will have a default plug associated with it.

As I have explained, when we navigate from one view to another view, we need plugs to travel from one view to other same uh if we consider this whole application as as one, uh as a one.

Broad picture, if I want to travel, if I'm to communicate from external view to internal environment, I I will need a plug through uh I will need a plug associated with my entire application which will.

Through which my external environment can communicate to the internal environment.

So that is nothing but there will there there there will be a need of an inbound plug to communicate in here with my window.

So in interface view will generate a default inbound.

Uh, so my when when I will create an web dynpro application, the application will communicate with interface view default inbound plug.

So this is how the external externally we are going to interact with your internal web dynpro application so.

That's the purpose of a component interface.

So it's nothing but an interface between V or external environment and internal environment.

So I will show you in the system and we will create an application how the default plug is created.

So this is not what we have to do it manually but but when we have to communicate between views, we have to define it on our own.

We have to define the plugs on our own but when we create an application.

To interact with so that our web input should interact with the browser at that time.

This interface we will by default create a default inbound plug.

Web dynpro application.

So once you're all you have worked all on audio nodes you have designed your views associated.

We have integrated it with Windows if put in that put all the logic in the controller.

Uh, and everything is ready.

So uh, as I said, to communicate it with the browser we have to create an application so that it should it should interact with your application.

So here we have created an application.

So an application is your entry point to a web dynpro component.

As soon as.

My mom, she can go to slide 16 again.

Yes, ma'am.

So just to relate this with a particular normal report program.

You can consider a component controller as a separate include where you have all the declarations so it is accessible to whole program.

Then we have custom controllers which maybe let's consider as you are creating a separate.

Set off.

Within a routine, maybe.

Let's consider that these declarations are within a routine.

Or maybe you are passing as a parameter.

So within 2 views you can pass the data using the custom controllers and the component controller is at a global level so if suppose you are reusing a program.

Then you can access the component controller declarations, but you can't access the routine declarations.

Then the interface view interface controller that you can consider it as a.

Let's say you are creating an LV program, so LV itself the the type pool of LV itself has a lot of declaration in that so.

That particular declaration, which is specific to an LV, that you can consider as an interface.

So the interface, let's say N is an LV.

It has its declaration and some routines and all to handle the maybe column width column that to be visible.

So those things are logical written.

Interface, controller, program global are in component, controller and program routines you can consider.

It as a custom controller just as this you can consider as a layman so.

If you are not having an if you cannot visualize a web dynpro component, you can visualize these controllers as a report program.

Thanks, Mike.

So after creating an app once the component is created so this is your Web and pro component.

So these are all the nodes associated to your web dynpro component.

Once the component is ready, we create a web dynpro application.

So application will generate a URL.

This URL you can use to access your application inside in the browser.

As I've explained, these automatically creates an inbound plug of which is of which is called default.

Inbound plug.

So your interface view will create a default inbound plug when you design your application.

All applications the URL associated with the special inbound plug in the interface view, so we will explain this to you in detail in the application itself.

So then we will design an end to end application.

So we will explain all how the default views is created and how we will.

Going to execute our application using the URL.

Web control components.

So these are the essential components of web dynpro.

One is window.

There is view controllers and component interface as we have seen in the architecture.

Uh, there's first thing in in your layout.

The first thing is which is going to be created as a window.

So window is the placeholder for all your view.

Use because view as an individual cannot be displayed so they need a place where they can be integrated into.

So they needs a placeholder.

So window is nothing but ur placeholder to all your views.

This is my window.

I have multiple views inside my window.

Every view has some UI elements.

Now this is a carrier.

This is an input field.

This is a label.

This is a button, so these are all input fields.

These are all UI elements.

Now your view will give you a large library of UI elements, so.

We will be looking into all those UI elements.

So there are N number of elements.

So we will be focusing on few of the elements which are commonly used like input field, labels, buttons, containers, tables.

So these are the most commonly used UI elements.

Now this view has a controller let's say this part.

In this particular view I entered some value in my input field and on a click of GO button I want to perform some operation.

So on this button action I would be putting any some logic so that button.

Action on that particular button will be.

The logic for that action will be written in the view controller.

So this view has its own view controller.

Now this is my another view where there's a kind of table or is an ALV display and click on this radio button I have to.

Read this particular line item. So the logic behind this will be written in the view controller of view 2. So every view has its own controller and all views are embedded together in A1 single window.

Not a.

This view has a visibility to hits its own controller and this view 2 has a visibility to its own controller.

I can't communicate from view 2 to view one controller or view one to view 2 controller, but the component which is your global controller, both views can access.

The the component, controller methods, events, or the context?

So from here they are explaining different view.

So I would say uh, I I'll go to see it and I'll show you how it exactly looks like.

So let me log into the system.

So this is my city.

So you will be opening your web in through application from sea transaction.

To select Webpro component slash interface from here.

Oh man, so I guess your SCT screen is not visible.

Can anyone else see that?

No, not not able to see.

Let me reshare myself.

I guess you have you have just shared your screen one.

So instead of that if we share desktop here.

Yeah, yeah.

Can you see it now?

Yes, yes.

OK so this is our transaction.

From the drop down I have selected web dynpro component and this is one of the demo application which I have created.

So today I'll be taking you through all these.

The different nodes of web dynpro and how they exactly look like in the application we will be looking at.

So once we create any application, the component, controller, interface, view and window, these four elements are created automatically automatically by the system.

The web dynpro application is the last thing after designing our entire application.

We create an application to.

Execute your web dynpro application.

Let's go to the component controller.

So this is this is my component controller.

This is the context area.

So while defining the done through VC, that's basically depends on the declarative.

Technologies that that means we whenever starting a web dynpro, we start with their declarations first and they.

Web improved design is all all controlled by what declarations we have done at your context area.

So in the context the.

When we design a new application it will give you a root node which is called a context node. So this is A1 single node will be created or automatically by web dynpro underneath the context node.

We will be designing our own nodes and attributes.

Now what are nodes and attributes?

Uh, so let's say uh.

Here I have given an example like uh.

I have created a structure here.

Uhm, let's say you want to create one variable, some flag maybe?

So what we will do is underneath the context context area, we will create an attribute.

So the two options, we can create a node or we can create an attribute.

So a node will be a collection of your attributes.

So it's like a structure where one or more attributes are associated together.

So the first thing we do is we create a node.

So so I have given.

The name of my node as sales detail.

Underneath that node I have created various attributes.

So these are all the variables of my structure.

So underneath the root that we have to always create all the custom nodes and attributes underneath the context root node given by web dynpro.

These these structures and variables which we have created in the context area will have a global visibility to all the views which we have created in the application.

Now in this example this is my note in in this particular node there are different attributes created like the BLE and the date time name.

So these are different elements.

These are different attributes associated to this node.

Now I have to.

Access this particular node in my view.

So every view will have its own context area 2.

The component controller context is a global context.

View has also has its own context which is which is local to this particular view.

So name of this view is a main.

View, so the main view context will be.

Visible to only main view.

If I create view 2, then view two will have its own context and the list of the attributes in the context of Main view will not be visible to the view 2.

Now when when I define some global declarations, when I made some global declaration in the component controller context and I want my view.

To uh have access to these contexts.

So we need to do some context mapping here.

So I want that in the right hand side you can see there's a component, controller, context uh picture given and this is your main view context.

This was the the.

Node I have created in my component controller and I want my main view to have access to.

A sales detail node created in the component controller now.

To make this view to make.

This uh this sales detail accessible at view level.

We need to tell this particular main view that yeah there is one sales detail node created app controller and you have access to this sales detail.

So how?

This main view will interact with the component controller view.

So what we have to do we have to map this particular sales details to the.

Context of main.

So what we do, we drag this sales detail from the controller and we drop it into the context of main view.

So what we are not creating new nodes or attributes at the view level.

We are just dragging the nodes created at the controller and dropping it into the context.

Of Main view, so nothing new is created just like the just the picture of the nodes at the controller and are now visible to the main view after dropping it into the context of main view.

So this is how we do all our declarations first and based on what node we need to access in the view we drag and drop it into the view.

So here there are two two nodes I want to display.

The sales detail in view one.

So I will drag this context of sales details into the view one context area.

And I want to display the sales details two node into second view then I will drag it from the controller and I will drop it into the second view so.

This is how we do the global declarations at the controller level.

Next to the contexts there are attribute.

So here we can define some global variables.

Uh, uh.

Uh like come let's say in view one we we have fetched some data and I want to store it into some global variable so that in view two I can access that that particular date.

So I can put create a global variable here, I can assign that data from view one into that global variable and I can use.

I can use that global variable in my view 2 because all the attributes which will be defining in the controller level will have visibility to.

Each and every view.

We will see how to use this existing context existing attributes like WD context and WD this when we will be starting with our programming.

So here we will in the events we will define events like an action on some button click setting the properties.

So different events will be defined at the event section.

And these are the methods.

So as as I said every controller has its own methods which is called hook methods.

So these are the the auto generated methods associated to component controller.

We will be learning each and every method, the functionality of each and every method soon as we will continue ahead.

For now just see there there are five such methods created.

These are not what I have created manually.

These are generated by system itself.

So there is a purpose of each and every method which will be learning and they are all executed in a particular sequence like.

If if you see do in it, it says a controller.

Initialization method, which itself explains that this is the method which is a.

Executed at first to initialize your component controller instance and you can see this do exit.

This will control all the cleanup methods.

So that means it will be executed at the end to do all the cleanup activity.

Once the application has been executed completely and you are closing your application, you want to do all the.

Clean up stuff.

You have to clean all your global variables.

You have to delete all your instances which you have created.

So all such activities will be performed at do exit.

So and then before navigation if you're navigating from one controller to another controller.

So if you want to perform certain action before navigating from one controller to others, will perform will do your logic in before navigation method.

So these are all different methods given by web intro.

And they have all their different, different purposes.

So this was this was all about a component controller, your master controller.

Uh, you will move ahead to the view, so.

If you if I open a view when you when you create your application, your web dynpro itself will generate 1 blank view and it will name it as main view.

You can also rename it later but it is mainly called as Maine.

So now you can find the few elements I have created here.

But when it was initially created it was a blank work area.

So this is a blank view created and this is a default view.

So when you will create your application and run your application, this main will be the default view which will be displayed in your browser.

Now let me show you where you can see that how how how I came to know that this is my default view.

As I said earlier that every view will be integrated into a window.

So if you open your window.

So here it's you can see the window structure.

This is your window.

And inside your window there is a view.

And you can see it is.

Well, I'm sorry you wanted to interrupt you.

I have paste one question into the meeting chat.

So whether the mapping or context mapping is done by?

Pass by value or pass by reference?

Uh yeah Jit, I was replying to that only.

OK so.

Basically when you are using a component very well in a component element in a particular view.

So if you are modifying that value at view level so it will reflect at component level as well or component level here yes as well.

So it's basically we can consider it as pass by reference only.

OK. Thank you.

Thank you.

So this this is our window structure.

So inside of window there's your main view there.

There's no need to add a main view manually into your window.

So our web dynpro itself will create this pattern.

It will self create view and add it into your.

Window and assign this main view as a default view.

So when I will create one more view, you'll find that it is not highlighted in yellow color.

Only the default view is highlighted in this particular way.

So this is here.

This is how you will come to know this is your default view.

Now, now let's go back to the view.

So as I've told you how to do context mapping, we will drag and drop your component controller node into your view context root node.

So now let's say.

Initially while designing of you have put up four to five fields in your sales details node and later the requirement comes that apart from these four to five fields I have also want to add to three more fields.

So I will go to component controller.

Then by right clicking on component controller I will add more attributes in my in inside my node.

Now this by doing this my component controller no no context is updated and now instead of three to four there are total 5 fields.

But still my view has only those three to four fields.

It is not yet reflected at my view level.

So there's one activity which we do to.

To reflect the changes in the component controller at the view level.

So either we can delete this node and drag and drop it again or we can right click on the node which is updated at the controller level and do an update mapping.

So what it will do it will adapt all the properties.

Of the controller.

Whatever things are updated at the controller level would be adapted by the view itself.

So that's the update mapping property which we do when when we make any changes at the node at context level.

So now how how we were going to create the node.

So let's let's try to create one node.

I will change this application.

So there are two options to create a node.

It can create it using a wizard or we can create it 1 by 1 manually.

So the first thing we'll create a node.

It's a test node.

You can see a new node kind of a folder is created.

So inside this node folder we will create our attributes similar to this.

So you can right click.

Create attribute.

Let's consider a VB.

So uh will be uh.

This is the data type of web.

Let me take the actual data type.

So now we will create one attribute of forebet field and it has an associated data type VBLN, under score, VA So what we will do?

We will right click on the node.

Create an attribute.

Let's name it as VBL only and the type will be the data element of of this particular attribute.

See this attribute is created at the bottom you will find the property of this attribute.

So this is your attribute name.

And the type of this attribute is blir, under score, VA.

So there are these are different other properties.

We'll see it 1 by 1.

Input field mode just automatic, so there are different input field modes.

This particular field has a.

The dictionary search help associated with it.

So if we keep it as automatic then it will adapt the dictionary searchable property and it will show you the search help for this particular field when you will execute the application.

So if we keep it as automatically it will automatically take the dictionary search self property if it if if this field has any associated with it or we can say we can deactivate these virtual property.

Or we can.

And give additional results are from DC we can design an obvious that is an object value selector search help or a freely programmable search app.

So these are different kind of search help options available at web dynpro, so.

Data dictionary we all know DIC search and if we select on DDI search and we can put the search help name in the search search help section.

So that search shelf will get associated with this particular field.

If I give it automatically it will automatically adapt the properties of that particular field or these are the two advanced options like object value selected or freely program.

So these these are the options which we will be covering in day four or win day.

Day three session, so I will keep it automatic for now.

Click on enter to refresh the page.

So this is the input field mode.

And uh.

If you want to keep these field readonly, don't want this to be an interactive field and you can assign this global property as readonly.

If you want to assign some default value to this particular field, you can give the the value inside the default value section.

Uh if if you want to uh if it's amount field you want to give it as a null value so you can put this checkbox as sticked.

So these are all properties associated to A and to attribute.

Now similar to an attribute, there are also few properties associated to no node.

So when you will click on node you will find these are the properties of a particular node.

So this is my node name so.

If it is an interface node, and that means it will be, this node will be.

Visible to the other applications which we are which are reusing this particular component.

Then import elements dictionary.

What is this dictionary structure so?

Here what we did, we created a blank node and then we are adding the fields one by one from manual leaf.

Some some table like I I was referring VBAK table.

What if I have to?

To create uh 10 to 15 fields or 10 to 15 attributes directly.

VBAK table so why?

Why should I do it manually 1 by 1 creating every single attribute.

So there's one option which we can do is we can with the help of code Wizard we can select the attributes from VBAK structure.

So let's.

Try to create one node using and.

We will.

What we will do will create a node.

Now here we will define from which structure we have to create your attributes from.

So VBAK is my structure and I will be creating all the attributes underneath my test two load from VBAK structure.

So now if you see the property of test two node, you can see there's a structure name VBAK.

Mention in the dictionary structure.

Now there is a button activated here which says from components of structure.

That means you want to transfer attributes from this particular structure into your node.

So we'll click here.

Now it has given me the list of all the fields which are in structure vrbka.

I want a end date, time, name and few some uh.

Feed amounts.

So these are the fields I these are the fields which will become attributes or which will become yeah which will become attribute underneath my test two node.

So I've selected what what all fields I need and I will click on continue.

You can see web dynpro itself has created all those attributes for me.

I don't have to manually go and create every each and every attribute 1 by 1.

So these are all now referring to your web Page Field data types.

So this is how we can refer to a particular structure.

So we can make your node referring to a structure and create attributes associated to that structure.

Now comes the cardinality of a node.

Now now every node has two types of cardinalities, one is a selection cardinality and one is a collection cardinality.

Now what does cardinality mean?

Now if if you have created a node that means this is this is a structure.

What what we can understood is this is my structure.

These are the fields of that structure and they are going to hold some data in the runtime.

Now I want that this particular node should hold only.

A single line item.

That means this should act like a structure, not a table.

That means it should collect only a single line item.

Which means the cardinality should be either one is to one.

Or zero is to one. So these are all several options available for cardinality. So zero is to 1 means it can have either 0 number of entries or Max 11. Line item if I have one is to one that means at least one.

Entry is necessary.

And can have maximum of 1 record.

So that means this test two node will act as a structure because it has a capacity of holding maximum one record.

If I make it.

So zero is to nor one is to.

It will mean that this particular node has a capacity of holding more than one records.

That means it is going to act like a table in your application.

So now the same node can act as a structure or as a table depending upon what cardinality you have given.

To this particular node.

So we called it as a collection cardinality, which is nothing but the capacity of the node to hold the data.

So either a single record or multiple records.

So for a single you'll go for.

One is 0 to one or one is to one and for multiple records we will go for zero is 12 or one is to north.

Now what is selection cardinality?

The cardinality was a collection cardinality and selection cardinal is how many number of records we can select from that particular node like from this particular if it.

If it is a structure that means I can either select zero records or one single record.

So if my cardinality collection cardinality is.

Zero is 2, one or one is 21, then the selection will be.

Will also be zeros to one or one is to one.

If the cardinality.

The collection cardinality is of Type 0 is 2 and or one is true and that means if it is a table then the selection can be a single line item selection or a multiple line selection.

So if I go if I select it as one is to nor zero is to.

Zero is 12 then my selection can be 10 to one or zero is to end anything. But if it is 0 to one then my selection cardinality cannot be one is 2 N because the collection is only one then how can how I will be selecting multiple?

So this is how they are depending dependent on each other.

So all we have to keep in mind is.

It will define the cardinality, will define the number of entries or number of data which a node can hold either A0 record or one single record or a multiple record.

So depending upon how your node is going to behave, how you want your node to to act like.

We have to decide the cardinality.

So for example for this node two in my view I have created a table.

So I want a I have.

I have binded my table UI element with node two node.

Then I want this node to be.

A to be of cardinality one is to end because it is behaving like a table in my.

Review but if it is a single line item in my view then I will make it as one is 2, one or zero is 2/1.

So this is how the the property will change depending upon how your UI looks like, how your design, your application design looks like.

Then there are few more properties associated to a new node which is a initialized, initialized lead selection.

This is uh, this is initially by default it is ticked.

You can untick this if you want.

So what what does this initialize lead selection?

So this mainly plays a role when you design a table, table or LV when your cardinality is basically zeros to end or one is to end.

So in case of table if you if we have set this lead selection or check.

Boxes ticked then.

Uh, it will the table.

The application will allow us to read only a single line from a particular table.

We can't select multiple line items from a particular.

So that is what is the that is what the property of lead selection is.

So when we will design a table UI element then at that time I will explain this in more detail how this particular thing work, but for now just giving you a broader.

View that this is something related to the data selection in your table or ALV.

These are these two properties.

This is a Singleton property and this is a supply function property.

So what is Singleton and supply function?

So when there is a requirement to.

Update your load order runtime so if there is a requirement to pre pre populate your load with some data.

So at that time we create a supply function so the we design method which is a supply function method and that method name is what we will be configuring in this particular area and the the, the.

There we will be putting some logic to pre populate our node date.

So it it depends on requirement.

Requirement how we we functionality is how we want to display uh data.

So I this will be this I will be covering in one of the demo example.

So it will be more clear that how the supply function in Singleton.

Uh, uh is used in the web delpro.

So for now, uh, it was.

I just wanted to give you a big picture that what is nodes, what is nodes and attributes, how they are associated to a root context node and how how we create them.

So we we saw that there are two two ways either create.

All the attributes 1 by 1 manually in underneath our node or order directly created from.

Sorry created from structure.

So this is how we create your contacts node.

And yes we we saw a few of the properties, common properties of node and attributes.

And the some complex properties like this we will be covering in in a few other demo examples.

So this was my context for component controller.

Now I will also show you as we have created these two new nodes underneath the context of component controller.

Now we can see how we will be using it in.

Our view?

Good one.

Any question?

Yeah. Hi, this is Sunil.

In the.

Actually in the node or text 2 so you have added the data dictionary structure VBAK correct.

So is there any provision can we add a multiple structure under that node?

So because yeah, what?

Yeah, so I have defined that this particular node will hold all the data from web to only.

So underneath this particular node, all the attributes will be associated to only VBA.

Let's say I I want to create a node with a combination of.

OK.

Two different table.

Means uh I want four fields from table one and three fields from Table 2.

So what I will do I will not, I will not define the structure type at the node level.

I will keep it this property as blank and underneath the my note I will.

I will create select the attributes using Wizard.

So I will.

I will.

Say underneath this node create an attributes from a particular structure.

So let me give you one example.

We will consider this test test node where we have not defined any DIC structure.

So now we have one field from VBAK table and now I will going to.

Add 2 fields from like eslite table.

So what I will do I will right click here using the wizard I will add the attributes from particular structure so instead of feedback and I will write F flight.

I will select few fields from S flight table.

So it has now 4 fields from S flight table and let's quickly show you.

Now add few more fields from Rebecca Table.

So if you now see this particular test node is a combination of attributes from two different structures.

Just because I have not given any predefined structure or type at the node level, I can add attributes from various different structures.

But in test two I have defined that it should only allow the attributes from VBAK, so it can only.

Uh, access the attributes of abke structure?

But after I have added the field from the VBA and from the data dictionary structure, I have removed the VBA and I had the EPO.

So can we do that like that?

Yes, we can do that.

So I have removed now the distribution structure from test 2.

Now I can add from.

Uh, attributes from different table, different structure.

Did you get that?

Yes, yes, I'm clear.

OK. Thanks.

So one more thing we can add here so.

Imagine right click on test 2.

Create using wizard.

So if you see there are two options here, attributes from a structure and then again there is a copy nodes of a different context.

So if you select this one.

So there's one more feature wherein you can copy.

A note or a particular attribute from some different component itself.

So even that you can do.

So maybe you can give the same component name.

And do I4 on the controller.

No, no.

On the no.

Let me give some.

No, no.

We can refer the same, same, same, same one.

So basically we can give any component name.

914.

It's not active.

Four legs.

Maybe I should activate it first and file?

No, no you're no no.

Your name is APR and you have given the component name as a RP.

It's why APR?

All right.

So if you see here you can see all the.

The components of this particular all the web applications.

So suppose you want to copy a particular node from one of the view or let's say component controller.

Just double click on that double click on component controller.

OK, and then you have option is to copy selected node or copy all.

Let's do uh, continue.

So here you can see all the nodes you are.

You can see all the nodes, you can just copy that.

So suppose you have created 2 components and again you need to create a node.

So instead of creating each and everything manually one by one, you can just copy that.

OK.

Thanks man.

Thank you.

So now we have seen how to create nodes in your component controller context.

Now we will see how we will.

Now we will see how to bind it with your view context.

We will have to stop after this binding.

OK, so so now I'm inside my.

My view if you can see at the right hand side you can see all the component, controller, context nodes.

Earlier there were only sales details and CS details two.

Now in the recent example we created test node and test to these two new nodes and in the context of my.

Main View these two nodes are still not visible.

So it so to make these two nodes iaccessible at my view level I have to do a context mapping.

So what I will do I will pick this test node.

I will drag this and I will drop it into the root node of view.

So it is creating a copy of test node from controller to view.

You can see now this node test node is being mapped with the controller.

So view, view context and.

Controller contacts are being mapped together in this particular folder.

You can see there is an arrow.

So this arrow is directing towards the controller context.

It indicates that this particular node is being copied and mapped from the main controller.

If you see this folder UI folder, it doesn't show any arrow.

It means that this node is created at the main view level.

It is not mapped from the controller area.

So I have manually created this UI node in my main view of context of main view.

I have not dragged and dropped it from any controller.

So all the nodes which are copied and mapped from the controller will have this particular symbol in their node folder.

So this is how we do drag and drop and.

Let's save this.

What we will do, we will add one more field in the test node.

Let's add these two fields.

Now we have two new new fields in the test load of of our controller.

I will go to my view.

Go to the context area of view.

Now open this test node at both the sides.

You can see we have amount and currency field at the controller level but they are not reflected at the view level.

So to update the binding of the test node to make it up to date as per the controller context we will right click on this node.

Undo update mapping.

Click on yes.

Now it has again copied all the updated attributes from the test node from component controller to the main view.

So you can see now the two new fields which we were we have added in the controller, now they are reflected in the view.

As well.

So this was about context mapping, how we define the contacts at the controller, how we access those contacts at the view level.

Uh, either we can map the context from the controller or we can design your own context at the view level similar to this.

So these are these are all and we have also seen the properties associated with node and the the individual attribute.

So are we good to close today's session, ma'am?

Is there any questions?

I see a question from former.

And she.

Yeah, if we directly make changes in the test mode or like if we add a variable, will it get reflected in the component controller?

So you you can't add.

As you see we are in, uh, change mode, but and I'm at my view level, so I can't add any attribute at the context of.

The test node attribute.

OK, if if I right click here it is not allowing me to add any attribute to the test node.

Because this test node is mapped from the controller, it is not created at view level.

So what we have to do, we have to go to the controller level, update it there and then do the update.

Mapping at the view.

But when we create nodes at the view level just like this UI node which is not at not which was not created at the controller level, it was designed at the view context.

So for this particular UI you can see this.

Uh, create nodes are active, so I can change.

I can modify only those nodes which are created at the view level.

And to modify the nodes which were created at the controller level, I have to move to the controller, update it there and then do the update.

Thank you.

Is it clear?

Any other questions?

So tomorrow we will be looking into how the UI elements work in view, how we will be designing our view and we will also see.

We will also take an example of creating multiple views and navigating between those.

Views using plugs, so that is what we will be covering tomorrow.

OK, so.

Thank you for joining.

A recording would be shared by the Beast.

This week.

Thank you in advance, it was a nice session.

OK.

Thank you.

Thank you for joining.

Thank you.