Modeling View Compositions in Web DynproJava

SAP NetWeaver 04

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Modeling View Compositions in Web Dynpro

You can download the Web Dynpro project for the current tutorial from the Software Developer Network SDN in two versions: one skeleton version you can use for exercises and one final version (solution) for an immediate build, deployment and run on the Java engine of the SAP Web Application Server.

Task

In this tutorial, you develop a Web Dynpro application that implements a complex view composition. The term view composition describes a set of all view assemblies that can be accessed by navigation. A view assembly consists of normal Web Dynpro views and interface views of Web Dynpro components within the browser window.

You can model view compositions in the Navigation Modeler as part of the Web Dynpro tools. You can use the following options to model view compositions:

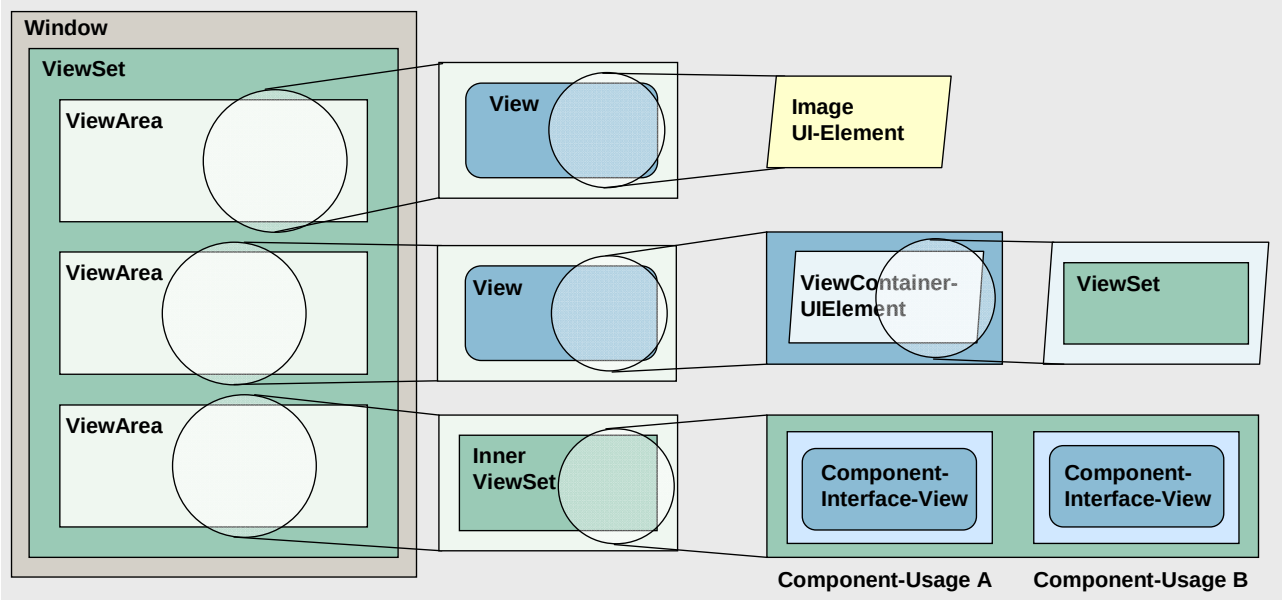
- Embedding several views in view sets. You define the division of these view sets into rectangular areas by selecting a particular layout (T layouts, GridLayout).
- Embedding view sets in each other.
- Embedding visible Web Dynpro components using their component interface views. In this tutorial, a Web Dynpro component is embedded using two instances.
- Embedding views and view sets in view layouts by using the *ViewContainerUIElement*.

To learn how to use these options, you develop a Web Dynpro application that displays three areas (view areas) in a browser window:

  - A graphic for displaying the implemented view composition using the *Image* UI element.
  - An area in which an inner view set is embedded using the *ViewContainerUIElement*. You want to execute a simple navigation change in the inner view set between a view and an EmptyView.
  - An area in which two instances (represented by two component interface views) of a Web Dynpro component can be controlled and displayed.

The schema below is a simplified representation of the view composition implemented in the example application.

Schema for the View Composition



Objectives

By the end of this tutorial, you will be able to:

✔	**Embed view sets in each other in the Navigation Modeler**
✔	Use the *ViewContainerUIElement* to embed views in an outer view
✔	Embed a view set in the *ViewContainerUIElelement*
✔	Use a Web Dynpro component twice (two component usages) and display it using the corresponding component interface views
✔	Control the lifetime of a component instance from the user interface

Prerequisites

Systems, Installations, and Authorizations

- The SAP NetWeaver Developer Studio is installed on your PC.
- You have access to the J2EE Engine.

Knowledge

- You already have experience of creating Web Dynpro applications - for example, by working through the Welcome Quickstart Guide (Creating Your First Web Dynpro Application).
- You have basic knowledge of Web Dynpro applications, view compositions and navigation, as explained in the tutorials Basic and General
- Knowledge of Web Dynpro components, as explained in the tutorial Using Server-Side Eventing in Web Dynpro Components, would be an advantage.

Next Step:

Importing a Project Template

Importing a Project Template

To restrict the development of our example application to the aspects set out in the objectives, a preprepared Web Dynpro project template is available in the SAP Developer Network (SDN) at http://sdn.sap.com.

The following steps take you through the procedure for enhancing this initial project template to a complete Web Dynpro project, which is also available to download separately.

- *TutWD_ViewComposition_Init.zip*: initial Web Dynpro project template
- *TutWD_ViewComposition.zip:* complete Web Dynpro project for view composition

Prerequisites

- You have a user ID and password to access the SAP Developer Network (http://sdn.sap.com).
- You have installed the SAP NetWeaver Developer Studio.

Procedure

Importing the Project Template to the SAP NetWeaver Developer Studio

1. Call the SAP NetWeaver Developer Network by using the URL http://sdn.sap.com and log on with the corresponding user ID and password. If you do not have a user ID, you must register before you can proceed.
2. Download the zip file **`TutWD_ViewComposition_Init.zip`** containing the initial Web Dynpro project *WDTut_ViewComposition_Init* and save the zip file to any local directory or directly in the work area of the SAP NetWeaver Developer Studio.
3. Extract the content of the zip file **`TutWD_ViewComposition_Init.zip`** in the work area of the SAP NetWeaver Developer Studio or to any local directory.
4. Call the SAP NetWeaver Developer Studio.
5. Import the Web Dynpro project *TutWD_ViewComposition*.

   For a detailed description of the procedure for importing Web Dynpro projects, see Importing a Project.
6. The Web Dynpro project *TutWD_ViewComposition_Init* appears in the Web Dynpro Explorer for further processing and completion of the tutorial.

Initial Project Structure

Once the Web Dynpro project template *TutWD_ViewComposition_Init* has been imported, the following project structure is displayed in the Web Dynpro Explorer:

Web Dynpro Project Structure
Web Dynpro project: **TutWD_ViewComposition_Init**
Web Dynpro application: **ViewCompositionApp** The application *ViewCompositionApp* displays the interface view of the Web Dynpro component *MainComp* in the browser window.

Web Dynpro component: **ImageComp**

A Web Dynpro component that is used twice by the outer component *MainComp*.

Views: **ImageView**

- The *ImageView* view displays a picture. The picture source and the title text displayed in the view are bound to calculated context attributes. The user uses the ImageComp component to transfer a parameter value (name of the component usage) to the component interface view controller. The parameter is required to calculate these context attributes . As a result, both uses of the Image component have a different appearance in the browser window.

Windows: **ImageComp**

The window only contains the *ImageView* view.

Web Dynpro component: **MainComp**

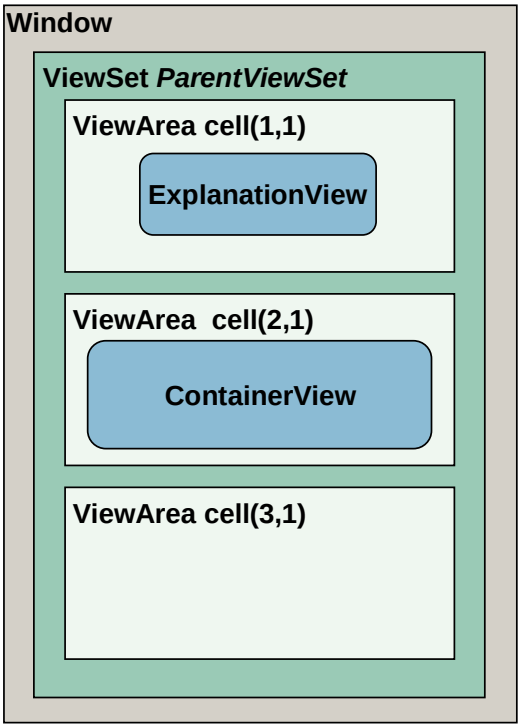
Views:

- **CompControlView**: View for controlling the lifetime of a component instance. A second instance of the same Web Dynpro component is displayed using the corresponding component interface view, or hidden by using EmptyView.
- **ContainerView**: View for embedding views and view sets by means of a *ViewContainerUIElement.*
- **ExplanationView**: The *ExplanationView* view displays the view composition modeled in the tutorial as a graphic, as it is displayed in the Application Modeler.
- **LeftView**: View embedded in the *ViewContainerUIElement*.
- **RightView**: Second view embedded in a *ViewContainerUIElement*.

Windows: **MainComp**

Initially contains a view set with three view areas (*ViewSetDefinition*: *GridLayout* with a column and three rows). The two views *ContainerView* and *ExplanationView* are already embedded in it.

In the Navigation Modeler, the initial Web Dynpro project is displayed schematically as follows:



The upper view area only displays a single view with a graphic.

In the ContainerView, the view composition is enhanced by a *ViewContainerUIElement*.

The view area that is still empty is for embedding an inner view set, in which two instances of a Web Dynpro component (*ImageComp*) can be displayed using component interface views.

Next step:

Developing the Example Application for a View Composition

Developing the Example Application for a View Composition

Process Flow

The following description of the example application for a view composition is divided into two theme areas:

1. Enhancement of the view composition using a ***ViewContainerUIElement***:
    a. You add a new *ViewContainerUIElement* to the predefined view layout of the *ContainerView* view.
    b. You embed an inner view set in the *ViewContainerUIElement* (contains two views and two navigation links).
    c. You learn more about using *ViewContainerUIElements*.
2. Enhancement of the view composition using a ***Component Interface View***:
    a. Introduction to embedding Web Dynpro components.
    b. You enhance the view composition in an inner view set of type *TLayout*.
    c. You display two instances of a Web Dynpro component by using two component interface views.
    d. You control the lifecycle of a component instance by using the interface `IWDComponentUsage` within the view controller implementation.

Next step:

Adding the ViewContainerUIElement

Adding the ViewContainerUIElement

In principle, you can position views within a view assembly by using either predefined view set layouts such as *GridLayout* or other *TLayouts* in the browser window (corresponds to a Web Dynpro window). You define the dimensions of the individual view areas at design time by using corresponding view set properties.

The *ViewContainerUIElement* gives you more freedom when positioning views in the browser window. You can embed this UI element in the layout of an existing view and position it by selecting suitable container layouts (*FlowLayout*, *GridLayout*, *MatrixLayout*, or *RowLayout*).

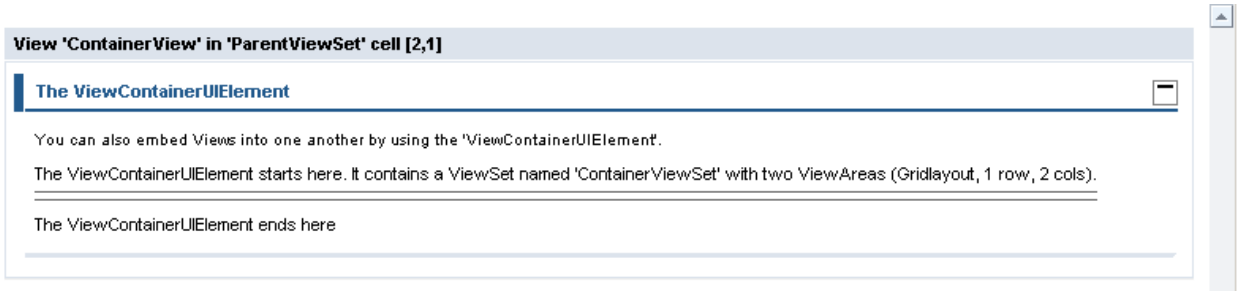
Whereas the layout-specific settings of the *ViewContainerUIElement* are made in the View Designer, you define its content in the Navigation Modeler. Thus, the content of the *ViewContainerUIElement* is not restricted to a single view, but can comprise any (inner) view composition. This means that you can also embed view sets in a *ViewContainerUIElement*.

Procedure

The following steps describe how to enhance the view composition of the initial Web Dynpro project by using a *ViewContainerUIElement*.

1. Expand the node TutWD_ViewComposition_Init → Web Dynpro → Web Dynpro Components → MainComp → Views and select the view ContainerView.

   The figure below shows how the view *ContainerView* looks in the View Designer:



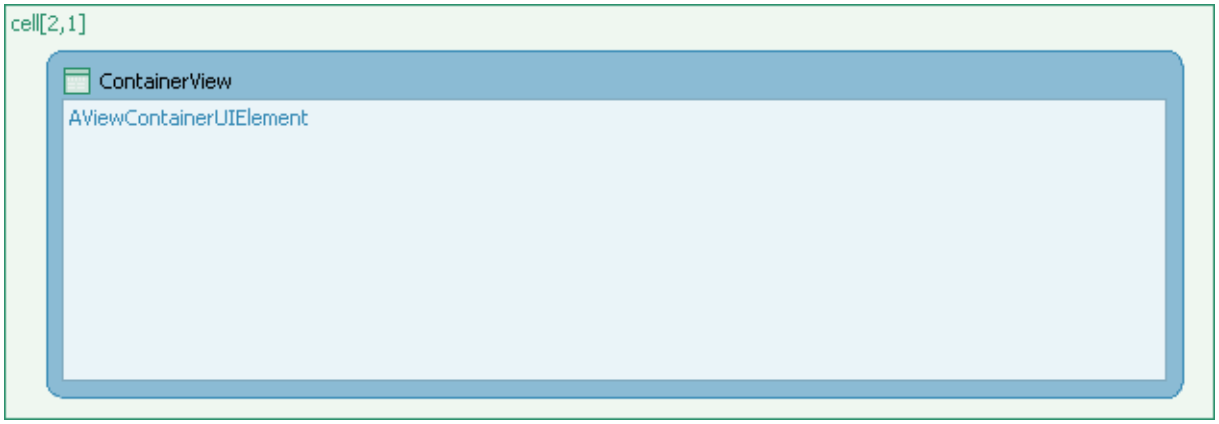
2. In the *Outline* perspective view, select the UI element *ATray* and then, in the context menu, choose *Insert Child*.
3. Add a *ViewContainerUIElement* called **`AViewContainerUIElement`**.

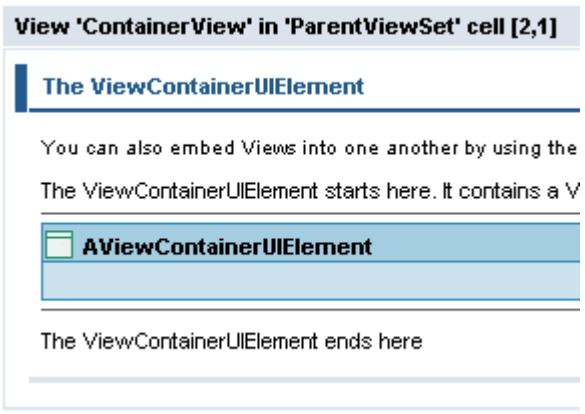
   You can embed the *ViewContainerUIElement* in a view in the View Designer and in the Navigation Modeler. In the Navigation Modeler, you do this by using the context menu entry *Create View Container*, which is available for all view nodes.
4. Set the property *Elementproperties of ViewContainerUIElement – layoutdata* to *MatrixHeadData*.
5. Move this UI element between the UI elements *StartViewContainerGutter* and *StopViewContainerGutter* using drag and drop (or the context menu entry *Move up*).
6. To save the new metadata, choose  (*Save All Metadata)* in the toolbar.

Result

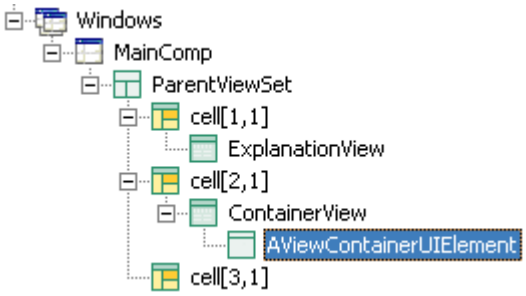
In the Navigation Modeler, the *ViewContainerUIElement* is displayed as an additional rectangle in the embedding view. It appears as an additional subnode in the Windows area of the Web Dynpro Explorer to enable you to model a view composition (view set with view areas, views, and navigation links) in the *ViewContainerUIElement*.



Display of a ViewContainerUIElement in the Navigation Modeler



View Layout



Web Dynpro Explorer

In the next step, you will enhance the new *ViewContainerUIElement* by adding a view set containing a view and an EmptyView.

Next step:

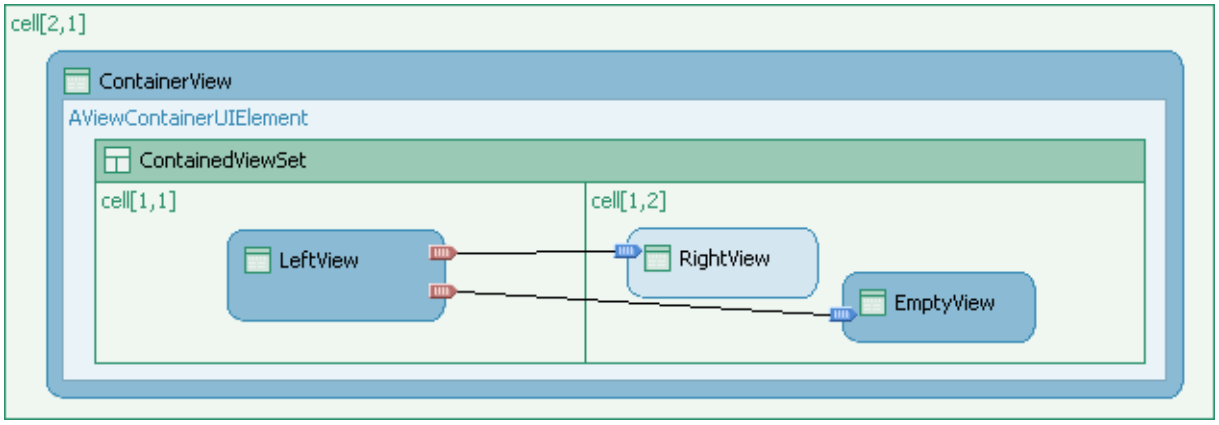
Embedding a View Set in a ViewContainerUIElement

Embedding a View Set in a ViewContainerUIElement

In the simplest case, the *ViewContainerUIElement* only contains a single view. It is possible to embed additional view sets; this special container UI element thus provides you with an additional flexible means of modeling view compositions.

In this step, you embed an inner view set in the ViewContainerUIElement *AViewContainerUIElement*. To be more exact, you enhance the existing view composition by adding those view assemblies that are modeled in the contained view set.

The initial project template already contains the two views *LeftView* and *RightView* for embedding in the inner view set.



Procedure

Embedding an Inner View Set

1. Open the Navigation Modeler for the window.
2. In the toolbar, choose *Create a viewset*.
3. Use the mouse pointer to bind a new view set called **`ContainedViewSet`** of type *GridLayout* into the *ViewContainerUIElement AViewContainerUIElement*.
4. In the Web Dynpro Explorer, choose the *ContainedViewSet* node.
5. Switch to the *Properties* perspective view and enter the value **`1`** for the *rows* property.
6. Enter the value **`50%`** for both properties *cellWidth[1,1]* and *cellHeight[1,2]*.
7. In the Navigation Modeler, adjust the position and size of the new view set.

Embedding Views in the Inner View Set

You can now enhance the view set that is contained in the *ViewContainerUIElement* by adding a view composition.

8. In the Navigation Modeler toolbar, choose *Embed a view*.
9. Embed the *LeftView* view in the *ContainedViewSet – cell[1,1]* view area.
10. Embed an *EmptyView* in the *ContainedViewSet – cell[1,2]* view area.
11. Embed the *RightView* view in the *ContainedViewSet – cell[1,2]* view area.

Defining Navigation Links

12. To finish, you complete the modeling of this view composition by declaring two navigation links. The project template already contains the corresponding inbound and outbound plugs for the *LeftView* and *RightView* views.

13. In the Navigation Modeler, declare the following navigation links. If necessary, enlarge the symbol of the *LeftView* view until both outbound plugs are visible.

Navigation Links			
Start View	**Outbound Plug**	**Target View**	**Inbound Plug**
LeftView	*ShowRightOut*	*RightView*	*NavigateToRightIn*
LeftView	*ShowEmptyViewOut*	*EmptyView*	*ShowEmptyView*

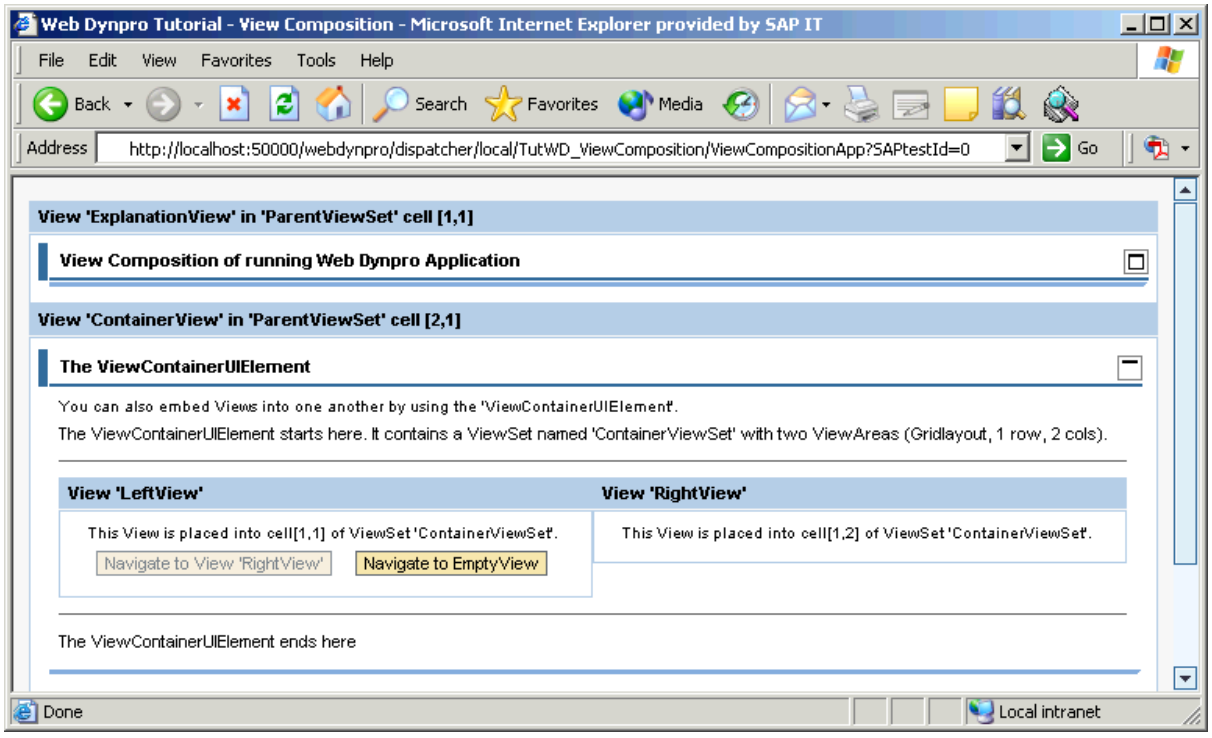
Starting the Application in a Browser

Once you reach this stage, you can start the example application in the Web browser and look at the status of your development.

14. Choose the node *TutWD_ViewComposition_Init → Web Dynpro → Web Dynpro Applications → ViewCompostionApp.*

15. In the context menu, choose *Deploy new archive and run*.

Result



The view assembly comprising the two views LeftView and RightView that is embedded in the *ViewContainerUIElement* in the *ContainerView* view is displayed in the browser window.

Go! Next Step:

More information about Using the ViewContainerUIElement.

Using ViewContainerUIElements

The *ViewContainerUIElement* is used mainly for two reasons:

- **View positioning:** Flexible design of view assemblies within the browser window.
- **Application performance:** The Web Dynpro runtime environment uses the *visibility* property of the *ViewContainerUIElement* to reduce the layout data sent to the client. Using the *ViewContainerUIElement* in this way can improve application performance.

View Positioning

Compared to the predefined view set layouts, the *ViewContainerUIElement* offers you additional options for designing view assemblies in the browser window. When positioning the *ViewContainerUIElement* in an embedding view, you can make use of the *FlowLayout, GridLayout, MatrixLayout*, and *RowLayout* layout types.

Rendering and Application Performance

Like all UI elements, the *ViewContainerUIElement* has the *visible* property, which controls its visibility within the view layout. The *visible* property can have one of the following three values: *none*, *blank*, and *visible*.

When view layout information is sent to the Web browser, the Web Dynpro runtime environment checks whether the current view assembly contains invisible *ViewContainerUIElements* (*visibility* equals *none,* either declared statically or by way of data binding to a context attribute of type *WDVisibility* and the *readOnly* property *true*). If this is the case, the embedded views and their UI elements are not rendered and thus not sent to the Web Browser.

Applications that need to use the *visibility* property in data binding to display and hide particular UI elements dynamically can define separate views for this purpose, which are then embedded in the view composition using the `ViewContainerUIElement`. The optimization approach is particularly useful when only one or two (of many) *ViewContainerUIElements* are to be visible in the individual view assemblies, as it reduces the round trip times considerably.

This optimization approach for reducing round trip times applies to the *visibility* property of the *ViewContainerUIElement*, but also to other *smaller* container UI elements such as the *Group*, *Tab*, or *Tray* UI element.

However, it is only supported when data binding of the *visibility* property to a context attribute of type *WDVisibility* and the *readOnly* property *true* takes place.

Next Step:

Using a Web Dynpro Component

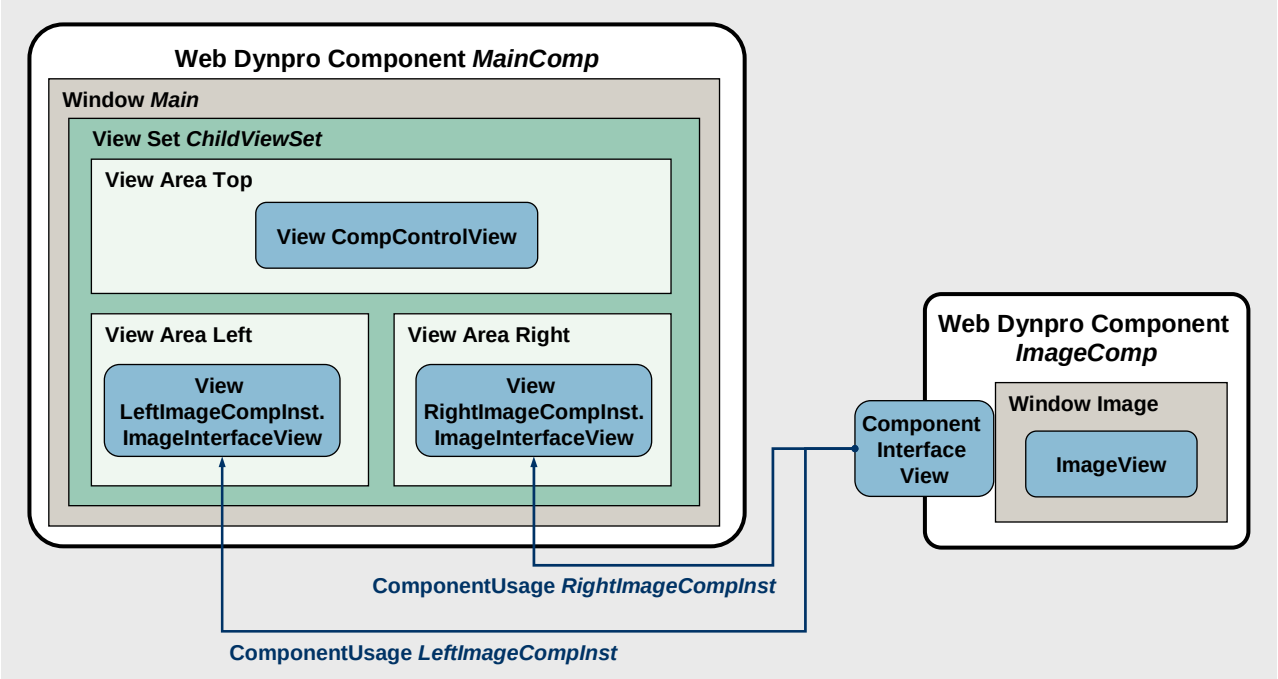
Using a Web Dynpro Component

In the second part of the example application, you enhance the existing view composition by adding an inner view set. A control view and two component interface views are displayed in this view set; to enable a single Web Dynpro component to be used in two instances, two component usages are declared.

For more information about embedding Web Dynpro components, see the tutorial Using Server-Side Eventing in Web Dynpro Components.

The following figure shows a graphical representation of the principle of visual embedding of Web Dynpro components. The image component is displayed using two independent instances. Each instance (component usage) is contained in the view composition using the corresponding component interface view.

For each window in the interior of a Web Dynpro component there is a corresponding component interface view for exterior use. Interface views are used like normal Web Dynpro views to embed Web Dynpro components within view compositions. A component interface view thus represents the abstraction of Web Dynpro components on the view level.



To model more complex view compositions, you have the option of embedding view sets in view areas in the Navigation Modeler. In the following step, you define a second view set called *ChildViewSet* and embed it in the *cell[3,1]* view area of the outer view set *ParentViewSet*. This inner view set is used to model the part of the total view composition that relates to the display and control of the two embedded component interface views.

Next step:

Embedding an Inner View Set

Embedding an Inner View Set

Procedure

In the initial project template, the view area called *cell[3,1]* is empty. In the next step, you embed an additional, inner view set in this lower view area to display the two component usages using corresponding interface views.

Embedding an Inner View Set

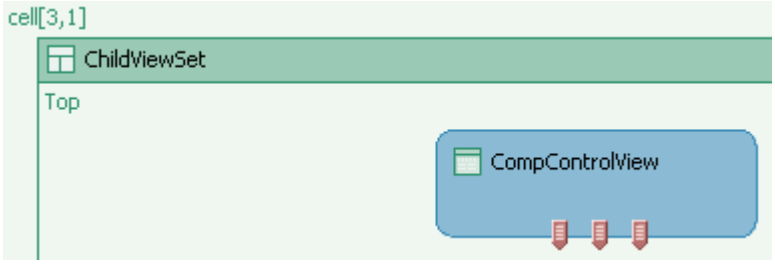
1. Choose the node TutWD_ViewComposition_Init → Web Dynpro → Web Dynpro Components → MainComp → Windows → MainComp.
2. Open the Navigation Modeler by double-clicking.
3. In the toolbar, choose *Create a viewset*.
4. Position the mouse pointer in the *cell [3,1]* view area and click the left mouse button.
5. Give the view set the name **`ChildViewSet`** and select the entry *TLayout* as the view set definition.
6. Make the following settings for the new view set:

Properties of the View Set *ChildViewSet*	
Property	**Value**
VerticalSashPosition	**`50%`**

Embedding the CompControlView View in the View Area

To control the display of the two component interface views, the project template already contains the *CompControlView* view. You embed this in the *Top* view area of the inner view set *ChildViewSet*.

7. You have opened the Navigation Modeler.
8. In the toolbar, choose *Embed a view*.
9. Position the mouse pointer in the *Top* view area and click the left mouse button.
10. In the dialog box that appears, select the *Embed existing view* radio button.
11. In the second dialog box, select the *CompControlView* view and choose *Finish*.
12. In the Navigation Modeler, enlarge the symbol for the *CompControlView* view that you just embedded until all three outbound plugs are visible:



Next step:

Embedding Component Interface Views in a View Set

Embedding Component Interface Views in a View Set

Procedure

To display two instances of the Web Dynpro component *ImageComp* in the embedding Web Dynpro component *MainComp*, you must first declare two component usages. You can then embed both instances of the Web Dynpro component *ImageComp* in the inner view set by using the corresponding component interface views.

Defining Component Usages in the Data Modeler

1. Choose the node *TutWD_ViewComposition_Init* → *Web Dynpro* → *Web Dynpro Components* → *MainComp.*
2. In the context menu, choose *Open Data Modeler*.
3. In the Data Modeler toolbar, choose *Embed an existing component*.
4. Position the mouse pointer in the *Used Web Dynpro Components* area and click the left mouse button.
5. In the dialog box that appears, make the entries shown in the table below and choose *Finish*.

Component Usages		
Name	**Used Web Dynpro Component**	**Lifecycle**
LeftImageCompInst	***ImageComp***	***manual***

   According to the Web Dynpro naming convention, component usages are named with the suffix **Inst** for *Instance*. The name of a component usage is different from the name of the corresponding Web Dynpro component in the same way that the name of an object instance differs from the name of the instantiated class.

6. Define the second component usage called *RightImageCompInst* in the same way, but with the lifecycle property *createOnDemand* instead of *manual*.

   Note the different selection of the *Lifecycle* property. The instantiation of used Web Dynpro components for *createOnDemand* is executed automatically by the Web Dynpro runtime environment as soon as the corresponding component interface view is visible for the first time in a view assembly. However, if the *Lifecycle* property is set to *manual*, this task is the responsibility of the application developer.

In the Web Dynpro Explorer, the defined component usages of the component *ImageComp* are represented by two additional nodes.

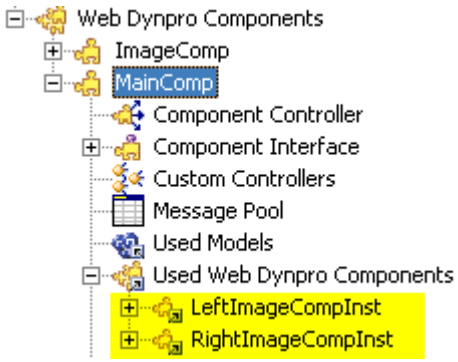
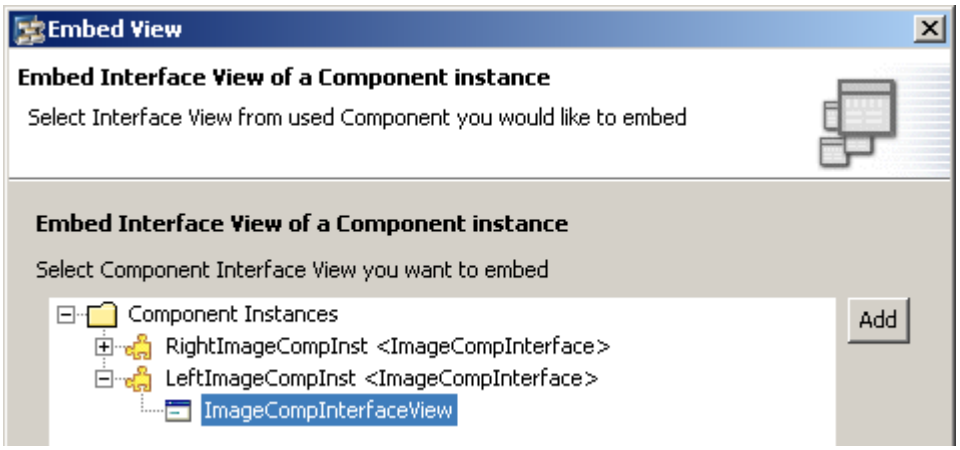


Figure 1: Display of Two Defined Component Usages in the Web Dynpro Explorer

Embedding Component Interface Views in a View Set

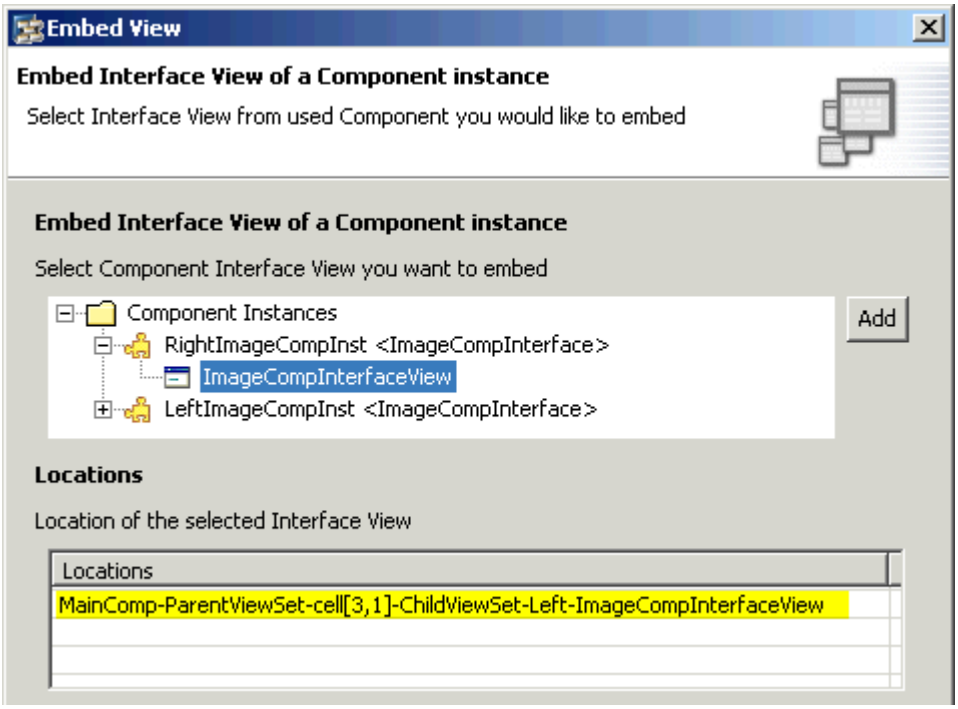
In this next step, you use the two defined component usages to enhance the existing view composition with the two corresponding component interface views.

7. You have opened the Navigation Modeler for the Web Dynpro window *MainComp*.
8. In the toolbar, choose *Embed a view*.
9. Position the mouse pointer in the *Left* view area and click the left mouse button.
10. In the dialog box that appears, select the *Embed Interface View of a Component instance* radio button and choose *Next*.
11. In the Embed View dialog box, choose *Component Instances* → *LeftImageCompInst* → *ImageCompInterfaceView* and choose *Finish* to confirm.



12. Repeat steps 8 to 10 to embed another component interface view, provided by the second component usage *RightImageCompInst*, in the *Right* view area.
13. In the Embed View dialog box, choose *Component Instances* → *RightImageCompInst* → *ImageCompInterfaceView* and choose *Finish* to confirm.

The *Locations* table displays all occurrences of a component interface view within the current modeled view composition.



.

14.

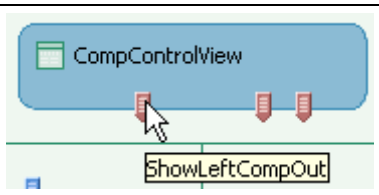
Defining Navigation Links

To complete the part of the view composition modeled in the inner view set, you use the next step to define three navigation links and also embed an additional EmptyView.

15. In the Navigation Modeler, add a new *EmptyView* to the *ChildViewSet – Right* view area. Then, in the *Properties* perspective view, assign the property default = *true* to this view.

16. In the Navigation Modeler, declare the following navigation links:

Navigation Links			
Start View	**Outbound Plug**	**Target View**	**Inbound Plug**
CompControlView	***ShowLeftCompOut***	***LeftImageCompInst. ImageCompInterfaceView***	***Default***
CompControlView	***ShowRightCompOut***	***RightImageCompInst. ImageCompInterfaceView***	***Default***
CompControlView	***ShowEmptyViewOut***	***EmptyView***	***ShowEmptyView***

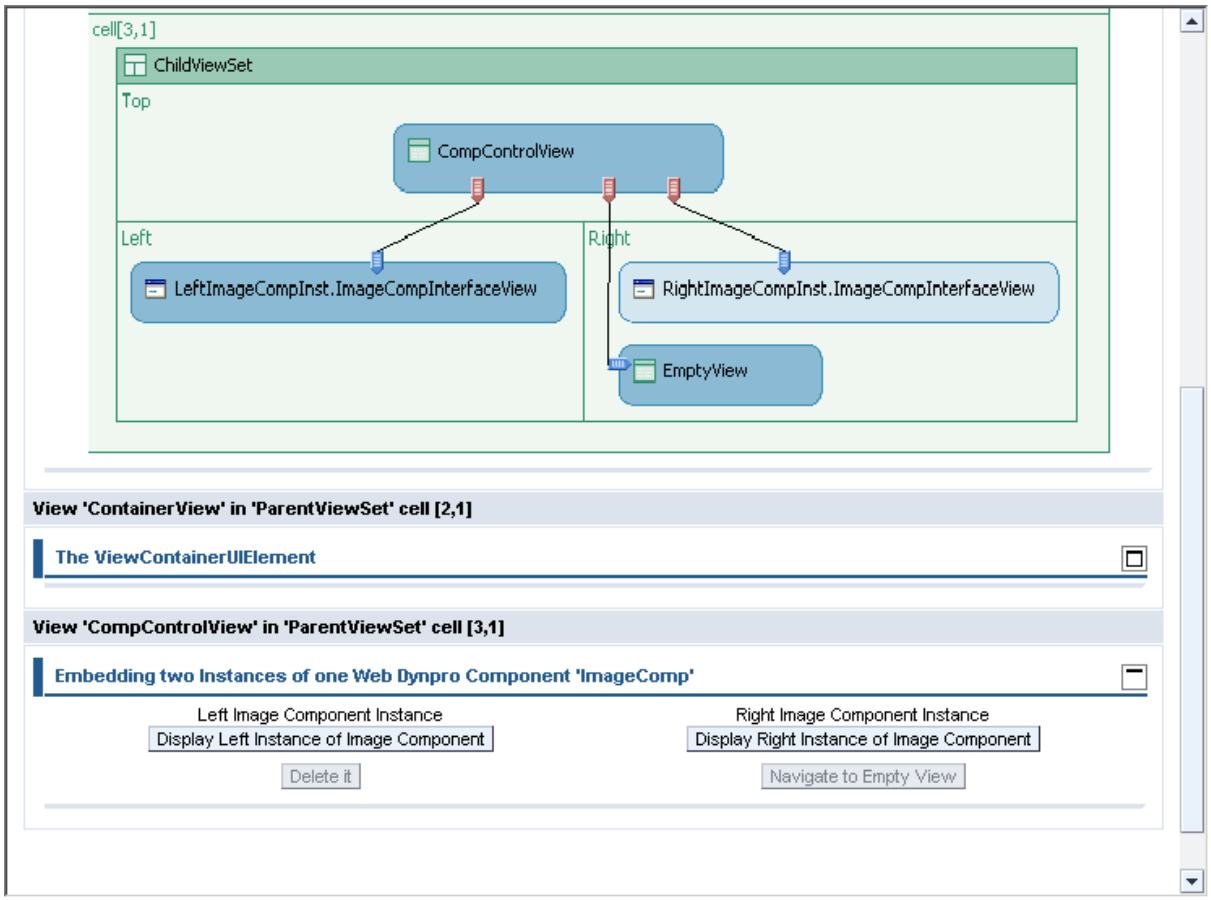
The plug name is displayed in the Navigation Modeler as a quick info.	

Result

You can now deploy the application and call it in the Web browser by calling the context menu for the node *TutWD_ViewComposition_Init* → *Web Dynpro* → *Applications* → *ViewCompositionApp* and choosing *Deploy new archive and run*.

The lower part of the browser window displays the *CompControlView* view, where you can create and delete the left component instance and display and hide the right component instance.

Remember that both component usages were embedded using different *Lifecycle* properties. Whereas the left underlying component usage is based on the *Lifecycle* value *manual*, the component usage belonging to the right interface view has the *Lifecycle* value *createOnDemand*. In the latter case, the image component is instantiated automatically by the Web Dynpro runtime environment as soon as the corresponding interface view is visible in the view assembly. Where *Lifecycle* = *manual*, the component interface view is only displayed in the view assembly when an active component instance exists for the component usage.



Next step:

Controlling the Lifecycle of a Component Instance

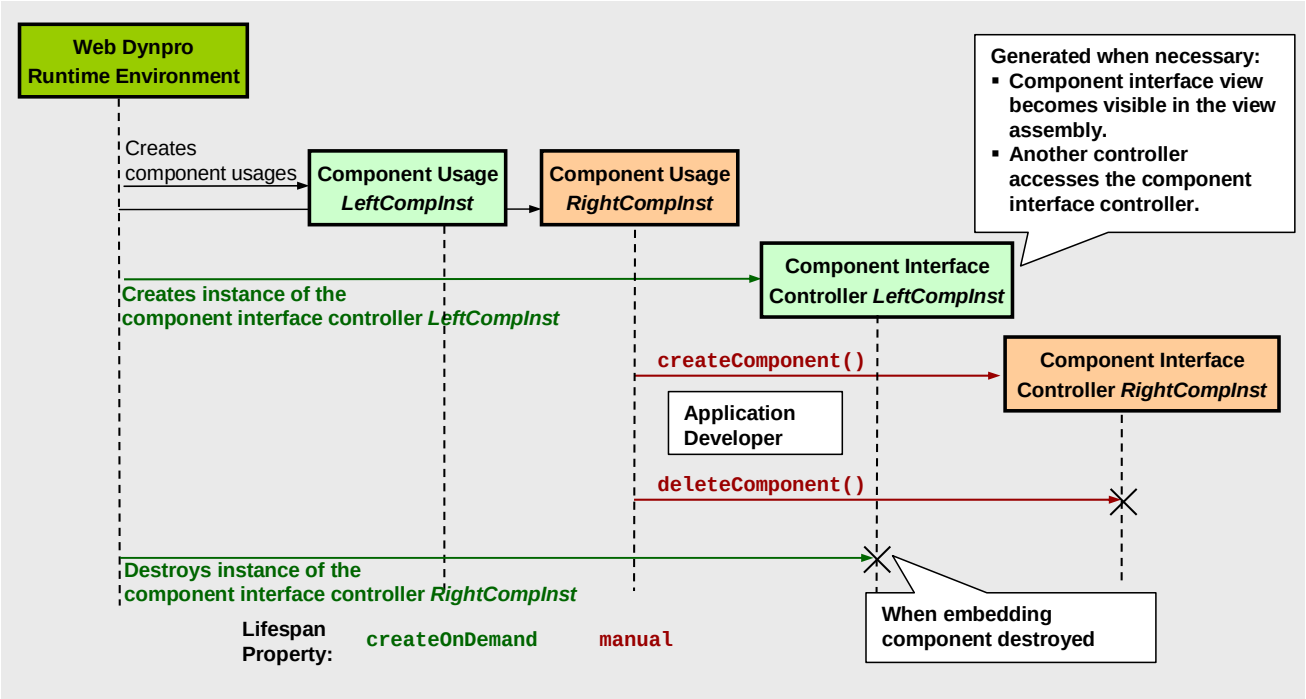
Controlling the Lifecycle of a Component Instance

In the final part of the tutorial, you enhance the implementation of the view controller `CompControlView.java` by adding source text lines to control the lifecycle of a component interface controller.

The lifecycle of a component interface controller is based on the *Lifespan* property of the corresponding component usage. This property can have the value *createOnDemand* or *manual*.

For component usages of type *createOnDemand*, the Web Dynpro runtime environment automatically instantiates the component interface controller (and visible components of the component interface view controller) when necessary.

For component usages of type *manual*, the application developer controls the component interface controller lifecycles using the `IWDComponentUsage` interface.



In our example application, the component usage *LeftImageCompInst* (the component interface view embedded on the left in the *Navigation Modeler*) has been defined with the *Lifespan* property *manual*. Accordingly, the control of the lifecycle of the corresponding component interface controller is the responsibility of the application developer. A component interface view contained in the current view assembly can only be displayed in the browser window if the component interface controller has been instantiated using the `IWDComponentUsage` method `createComponent()`.

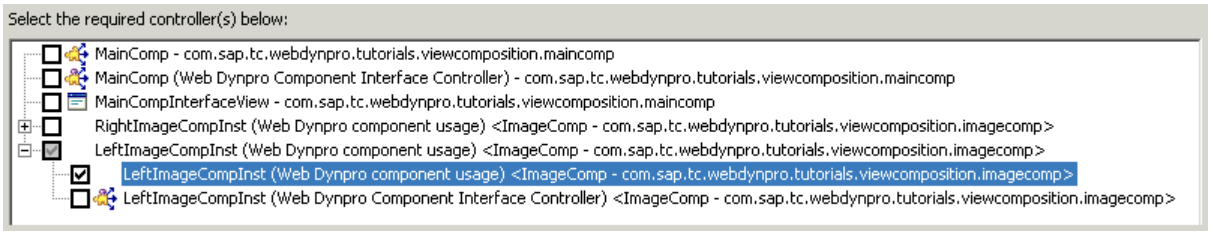
Procedure

Defining the Use of the Component Usages

To be able to access the component usage *LeftImageCompInst* in the controller of the *CompControlView* view, you must declare the corresponding use in the *Properties* perspective view.

1. In the Web Dynpro Explorer, choose the node *TutWD_ViewComposition_Init → Web Dynpro → Web Dynpro Components → MainComp → Views → CompControlView.*

2. Switch to the *Properties* perspective view. Under *Required Controllers*, add the uses of the component usage *LeftImageCompInst*.



3. Switch to the *Implementation* perspective view of the *CompControlView* view.

Creating a Component Instance

4. To create the component instance displayed on the left in the view controller class `CompControlView.java`, implement the following source text. The instance is not created until the user chooses *Create Left Image Component* in the browser window. This action is handled by the method `onActionCreateLeftImageComponentInstance()`in the view controller.

```
...
//@@begin javadoc:onActionCreateLeftImageComponentInstance(ServerEvent)
/** Declared validating event handler. */
//@@end
public void onActionCreateLeftImageComponentInstance(
  com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent )
{
  //@@begin onActionCreateLeftImageComponentInstance(ServerEvent)

  // because we set the 'lifecycle' property of the left component usage
  // as 'manual' the corresponding component instance must be created here
  wdThis.wdGetLeftImageCompInstComponentUsage().createComponent();

  wdThis.wdGetCreateLeftCompInstanceAction().setEnabled(false);
  wdThis.wdGetDeleteLeftCompInstanceAction().setEnabled(true);

  // navigate to the left Component Interface View, pass name of the
  // component usage
  wdThis.wdFirePlugShowLeftCompOut("LeftImageCompInst");

  //@@end
}

...
```

Destroying a Component Instance

5. To terminate the lifecycle of the component instance created earlier, the interface `IWDComponentUsage` provides the method `deleteComponent()`. Add the missing source text line to the method `onActionDeleteLeftImageComponetInstance()`:

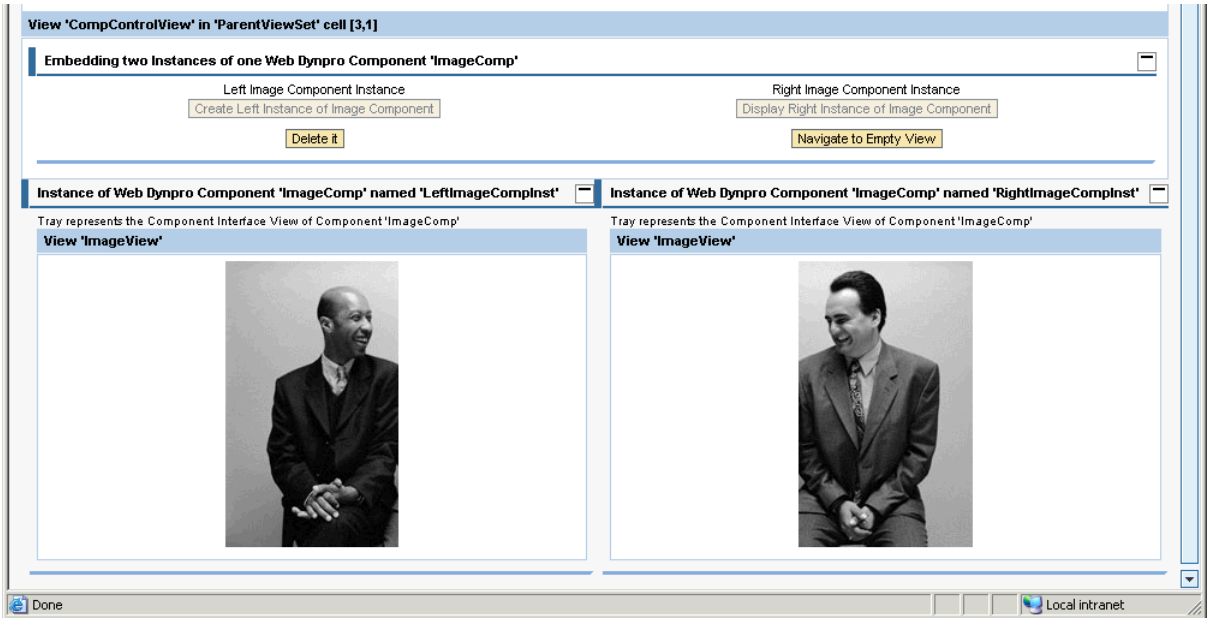
```
...
//@@begin javadoc:onActionDeleteLeftCompInstance(ServerEvent)
/** Declared validating event handler. */
//@@end
public void onActionDeleteLeftCompInstance(
  com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent )
{
  //@@begin onActionDeleteLeftCompInstance(ServerEvent)

  // delete left instance/usage of Component ImageComp
  wdThis.wdGetLeftImageCompInstComponentUsage().deleteComponent();

  wdThis.wdGetDeleteLeftCompInstanceAction().setEnabled(false);
  wdThis.wdGetCreateLeftImageComponentInstanceAction().setEnabled(true);
  //@@end
```

```
}
...
```

Result

In the context menu of the node *TutWD_ViewComposition_Init → Web Dynpro → Applications → ViewCompositionApp*, choose *Deploy new archive and run* to call the application and redeploy it in the Web browser.

After creation of the left instance of the Web Dynpro component *ImageComp* and navigation to the interface view of the second instance, the browser window displays the following screen:

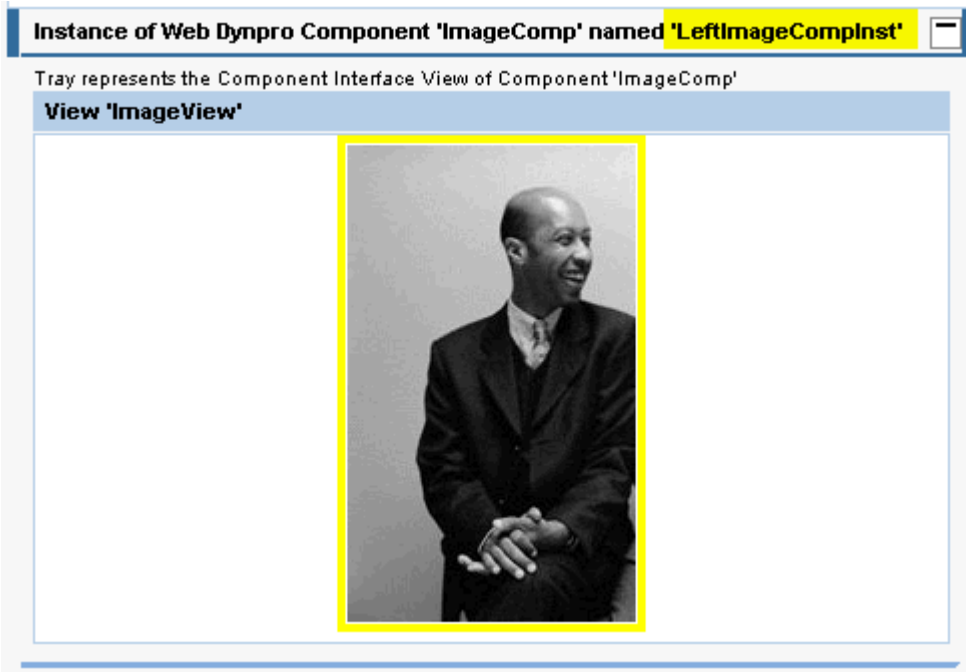


Whereas you can create and destroy the first component instance (displayed on the left) by choosing *Create Left Instance of Image Component* and *Delete it*, respectively, in the second instance (displayed on the right), you can only navigate between the component interface view and the empty view.

Once you reach this stage, you have successfully completed the tutorial for modeling view compositions.

The final step is to briefly explain why the two component interface views display different contents (title texts, picture sources) in the example application.

Why Do the Two Interface Views Display Different Contents?



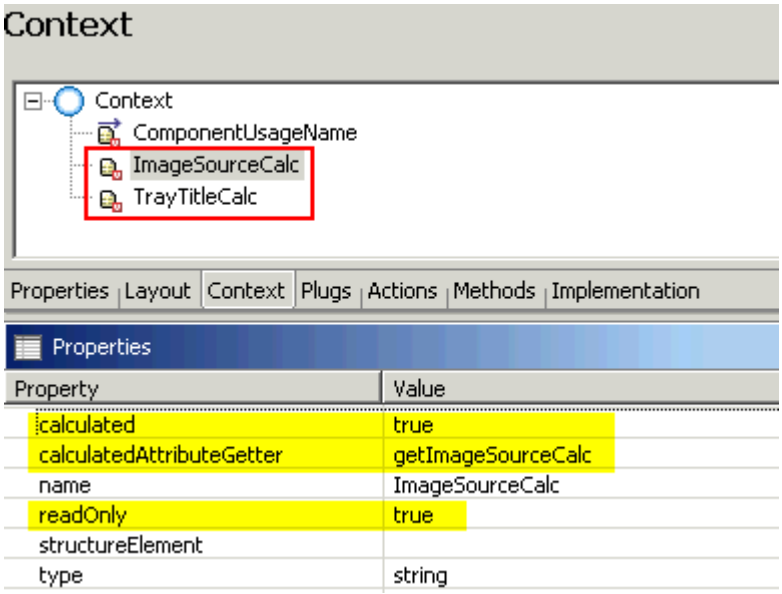
You will notice that the two component interface views in the browser window display different contents. Firstly, the view titles contain the name of the underlying component usage, and secondly, two different pictures are displayed. The reasons for this are as follows:

1. **Parameter transfer using inbound and outbound plug:** In both cases, the method `wdThis.wdFirePlugShow<Left or Right>CompOut(java.lang.String` **`name`**`)` is called to navigate to the display of a component interface view.

   The string parameter **`name`** (name of the component usage) transferred from the outbound plug is received by the inbound plug event handler in the component interface view controller:

   ```
   public void onPlugDefault(
     com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent,
     java.lang.String name )
   ```

   Once this string value has been saved in the context of the component controller, the name of the underlying component usage is available to the view controller by means of context mapping.
2. **Calculated context attribute:** In the context of the *ImageView* view, two context attributes are declared as *calculated context attributes* of type *readOnly=true*:



This means that the value of these two attributes is not saved in a separate value attribute, but is calculated in the view controller when necessary (the value is required for data binding to display a view).

```
//@@begin javadoc:getTrayTitleCalc(IPrivateImageView.IContextElement)
/**
 *  Declared getter method for attribute GroupTitleCalc of node Context
 *  @param element the element requested for the value
 *  @return the calculated value for attribute GroupTitleCalc
 */
//@@end
public java.lang.String getTrayTitleCalc(IPrivateImageView.IContextElement
element) {
  //@@begin getTrayTitleCalc(IPrivateImageView.IContextElement)
  String groupName =
    "Instance of Web Dynpro Component 'ImageComp' named '"
      + wdContext.currentContextElement().getComponentUsageName()
      + "'";
  return groupName;
  //@@end
}

//@@begin javadoc:getImageSourceCalc(IPrivateImageView.IContextElement)
/**
 *  Declared getter method for attribute ImageSourceCalc of node Context
 *  @param element the element requested for the value
 *  @return the calculated value for attribute ImageSourceCalc
 */
//@@end
public java.lang.String getImageSourceCalc(
  IPrivateImageView.IContextElement element) {
  //@@begin getImageSourceCalc(IPrivateImageView.IContextElement)
  if (wdContext.currentContextElement()
            .getComponentUsageName().equals("LeftImageCompInst")) {
    return "leftperson.jpg";
  } else {
    return "rightperson.jpg";
  }
  //@@end
}
```

In this way, the correct title text and a different picture source are calculated according to the underlying component usage.