How to Add UI Elements Dynamically in Web Dynpro

Applies to:

Web Dynpro for Java UI development, SAP NetWeaver 2004.
For more information, visit the User Interface Technology homepage.

Summary

This article briefly describes how to create User Elements dynamically in web dynpro for java and how create context and bind them to dynamically created UI elements.

Author: Arnab Pal

Company: TATA Consultancy Services

Created on: 21 July 2009

Author Bio

Arnab Pal is a BE in Information Science and engineering. He is working with Tata Consultancy Services for nearly 2 years. He is working in SAP Enterprise portal for 1 year mainly in Java Web Dynpro programming.

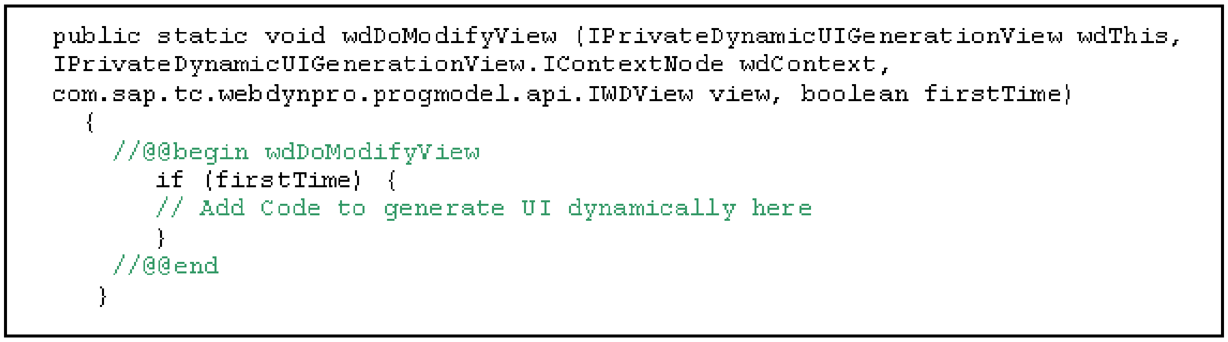
Table of Contents

Scenario

There may be a situation when the number of UI elements to be added in the view may not be known during design time i.e. a context of unknown structure. In such cases the UI elements are to be generated during runtime. However it is preferred that Dynamic generation of user interface elements, context attributes if these are not known at design time since this is might affect the performance of the program.

Creating UI elements at Runtime

Creation of user interface elements during runtime only possible from within the hook method *“wdDoModifyView ()”* which is present in the Web Dynpro view controller.



```
public static void wdDoModifyView (IPrivateDynamicUIGenerationView wdThis,
IPrivateDynamicUIGenerationView.IContextNode wdContext,
com.sap.tc.webdynpro.progmodel.api.IWDView view, boolean firstTime)
  {
   //@@begin wdDoModifyView
      if (firstTime)  {
      // Add Code to generate UI dynamically here
      }
   //@@end
   }
```

The following Parameters are related with *wdDoModifyView*:

- ***wdContext*** :It is used to access the root context node element of the view Controller
- ***wdThis****:* It is used to access the context of the controller, interface of the view controller and also to trigger plugs , events and access action objects.
- ***view:*** It is used to refer the layout of the view controller.
- ***firsttime****:* It is a Boolean type variable. It is true as long as the *wdDoModifyView* is called for the first time during the lifecycle of the view controller.

Implementation of *wdDoModifyView()* to generate UI Elements dynamically

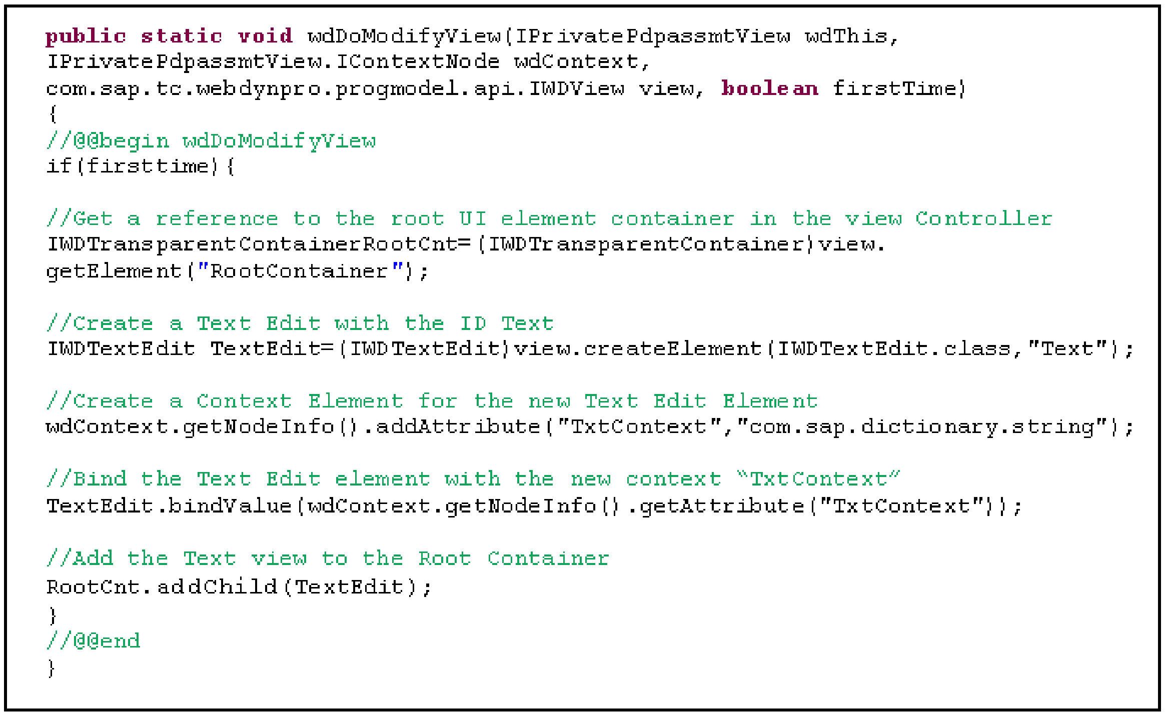
Every UI element must have a layout manager assigned to it. The layout manager defines how the child UI elements will be arranged with the parent container. In the following example we will see how a new element is created, a new instance of layout manager is assigned to it and how the UI element is added to the root container.

```
public static void wdDoModifyView(IPrivatePdpassmtView wdThis,
IPrivatePdpassmtView.IContextNode wdContext,
com.sap.tc.webdynpro.progmodel.api.IWDView view, boolean firstTime)
{
//@@begin wdDoModifyView
if(firsttime){

//Get a reference to the root UI element container in the view Controller
IWDTransparentContainer RootCnt=(IWDTransparentContainer)view.
getElement("RootContainer");

//Create a Text view with the ID Text
IWDTextView Text=(IWDTextView)view.createElement(IWDTextView.class,"Text");

//Ceare an Instance of Layout for the new UI element
IWDGridData TextGridData=(IWDGridData)Text.createLayoutData(IWDGridData.class);

//Set the different properties of the UI layout
TextGridData.setColSpan(1);
TextGridData.setHAlign(WDCellHAlign.LEFT);
TextGridData.setVAlign(WDCellVAlign.TOP);
TextGridData.setPaddingBottom(IWDGridData.DEFAULT_PADDING_BOTTOM);
TextGridData.setWidth("20%");

//Set the design properties of the New text view
Text.setDesign(WDTextViewDesign.EMPHASIZED);
Text.setTextDirection(WDTextDirection.LTR);
Text.setWrapping(true);
Text.setText(“Dynamically Created Text View”)

//Add the Text view to the Root Container
RootCnt.addChild(Text);
}
//@@end
}
```

In the above code module a TextView Element was dynamically created from within the wdDoModifyView(). An instance of the layout manager was created and assigned to the new TextView Element. Different Layout properties of the element were set. Then the new TextView element was added to the root container.

Binding Dynamically Generated UI Elements with Context Element

Some of the dynamically generated UI element might require to be bind with a context element for proper functioning like InputFiled , drop downs ,radio buttons etc.The following example shows how to create a TextEdit UI element and bind it with a context element.

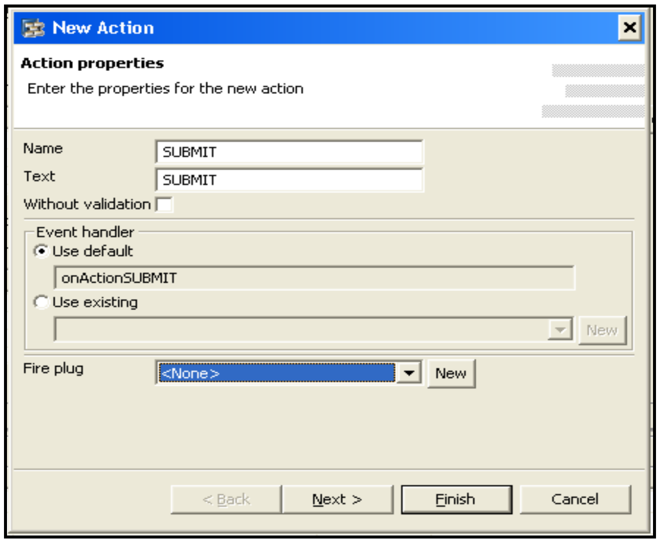


```
public static void wdDoModifyView(IPrivatePdpassmtView wdThis,
IPrivatePdpassmtView.IContextNode wdContext,
com.sap.tc.webdynpro.progmodel.api.IWDView view, boolean firstTime)
{
//@@begin wdDoModifyView
if(firsttime){

//Get a reference to the root UI element container in the view Controller
IWDTransparentContainerRootCnt=(IWDTransparentContainer)view.
getElement("RootContainer");

//Create a Text Edit with the ID Text
IWDTextEdit TextEdit=(IWDTextEdit)view.createElement(IWDTextEdit.class,"Text");

//Create a Context Element for the new Text Edit Element
wdContext.getNodeInfo().addAttribute("TxtContext","com.sap.dictionary.string");

//Bind the Text Edit element with the new context "TxtContext"
TextEdit.bindValue(wdContext.getNodeInfo().getAttribute("TxtContext"));

//Add the Text view to the Root Container
RootCnt.addChild(TextEdit);
}
//@@end
}
```

Note: In the above code module the TextEdit element was created dynamically and a context "TxtContext" was also created from within the wdDoModifyView() and was binded with the new element. However it is preferred to create the Context Elements outside the wdDoModifyView () inside some other Function module as the wdDoModifyView() should be used for alteration of the view attributes not data attributes.

Creating Action for Dynamically Generated UI Elements

Some of the UI elements may require actions associated with them for proper functioning as in case of a Button. As Event handlers cannot be created dynamically, Event handlers are created during design time only and then associated with dynamically created action of the corresponding UI elements. The below section shows an example of an event handler being created during design time and then it is associated with the action of a dynamically created Button UI element.



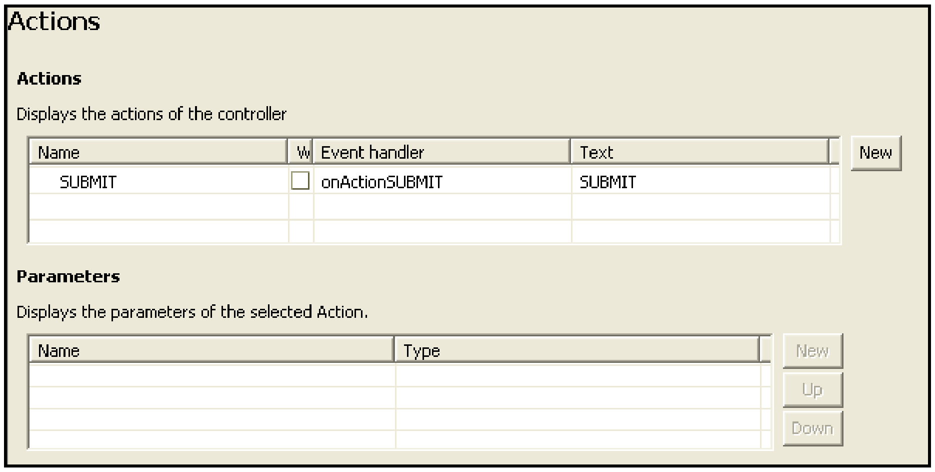
Action properties

Enter the properties for the new action

Name	SUBMIT
Text	SUBMIT
Without validation	☐

Event handler
[✓ Use default] onActionSUBMIT
Use existing

Fire plug	<None>



Actions

Actions

Displays the actions of the controller

Name	W	Event handler	Text
SUBMIT	☐	onActionSUBMIT	SUBMIT

Parameters

Displays the parameters of the selected Action.

Name	Type

The above figure shows that an action has been created with an event handler during design time only which will be assigned to the dynamically created button.

```
public static void wdDoModifyView(IPrivateInnerCompView wdThis,
IPrivateInnerCompView.IContextNode wdContext,
com.sap.tc.webdynpro.progmodel.api.IWDView view, boolean firstTime)
  {
    //@@begin wdDoModifyView
      if(firsttime){

      //Get a reference to the root UI element container in the view
      //Controller
      IWDTransparentContainer RootCnt=(IWDTransparentContainer)view.
      getElement("RootContainer");

      //Dynamically Create a Button
      IWDButton button =
      (IWDButton)view.createElement(IWDButton.class,"Button");

      //Dynamically Create an Action and link with the Action Event
      //Handler created during design time
      IWDAction btnActn= wdThis.wdCreateAction
      (IPrivateDynamicView.WDActionEventHandler.SUBMIT,"Submit");

      //Set Action for the dynamically created Button
       button.setOnAction(btnActn);

      //Add the Button view to the Root Container
      RootCnt.addChild(button);
      }

    //@@end
  }
```

The above code module shows that a button has been dynamically created and action has also been created for the corresponding button. The dynamically created action has been set with the Event handler created during design time. Now the programmer can write what ever code he wants within that event handler during design time only as show below.

```
public void
onActionSUBMIT(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent
wdEvent )
  {
    //@@begin onActionSUBMIT(ServerEvent)

            //Code to write for dynamically created Button

    //@@end
  }
```

Points to Remember while Using *wdDoModifyView()*

Though one may find using the functionalities of the wdDoModifyView() very easy to play around with UI design of the view controller it is highly recommended not to use dynamic UI construction as it will introduce performance overheads. Following are the few points (among many recommended by SAP) to keep in mind while doing dynamic UI generation.

- Dynamic UI design should be done only when the maximum no of UI elements, or context elements are not known during design time.
- As little as possible coding should be done with in wdDoModifyView().
- The context nodes and attributes required for the dynamically created UI elements should be built from outside the wdDoModifyView() preferably either during design time or before entering the wdDoModifyView() module.
- The modification of UI elements should be done when the Boolean parameter **firsttime** is true. Doing the coding after it becomes false will mean there will be round trip every time there is an user interaction giving rise to performance overhead.
- After creation of the UI element it should be bind with a Context element and any further modification of the UI element should be controlled by manipulating the Context.

References

https://www.sdn.sap.com/

http://help.sap.com/

"Inside Web Dynpro for Java" – Chris Whealy

For more information, visit the SOA Management homepage.

Disclaimer and Liability Notice