

# ABAP WebDynpro Tips for Beginners

So you've done the ABAP Webdynpro course, and you've maybe built a few simple webdynpros. Perhaps you now need to build something rather more complex, and what you learnt on the course doesn't seem to answer all your questions? Having recently been through that situation myself, in this blog I hope to pass on a few tips that might help you.

I will try to avoid repeating information that's already out there, so don't treat this as a complete development guide - remember to look at other SDN posts, SAP Notes and the online help as well.

I intend to keep my tips fairly brief - I'm aiming to give pointers rather than a step-by-step guide. But over time I may be able to expand on some of it - in response to your feedback and as I continue to learn myself. I have provided links to the relevant sections in SAP Help for most topics. So let's get started...

## Windows and Views

I was confused about these when I started my development. Did I need a window for every view? Basically:

- a window is a collection of related views.
- navigation between the views is defined in the window.

For example your application may have an initial selection view, a report view and a details view. The user should see only one view at a time, and they should all be shown fullscreen. That's three views, and you would require a single window to define navigation between them.

For clarity, it may be useful to have a naming convention to distinguish between windows and views when programming. For example you could start view names with 'V\_' and window names with 'W\_'.

A situation where a second window may be needed is for popups.

## Popups

In the example above, it could be that you want the details view to appear as a popup. In this case a second window is required for the popup.

Note: in the web dynpro code wizard there's an option to create and open a popup - you just need to provide the window name.

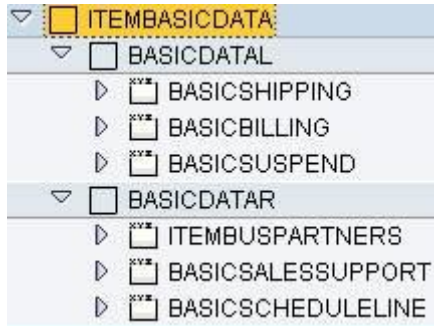
[http://help.sap.com/saphelp\\_nw70/helpdata/en/43/bcd2b8e326332ee10000000a11466f/content.htm](http://help.sap.com/saphelp_nw70/helpdata/en/43/bcd2b8e326332ee10000000a11466f/content.htm)

See example component WDR\_TEST\_POPUPS\_RT\_00.

## View Layout

Of the three types of view layout available, I generally favour using Matrix layout - I found this easy to use and it produces tidy-looking screens. However the other layouts will have their uses also - for example I may use Flow layout to put 'from' and 'to' fields next to each other when showing a range.

I also make extensive use of transparent containers in my view layouts:



In the example above, I've divided my 'item basic data' fields into two columns by using transparent containers - I then have three groups of fields in each column. You can set the type of layout independently in each container. So it is quite possible to have a view where some parts have a Matrix layout, and other parts have a Flow or Grid layout.

In my 'ITEMBASICDATA' container, I've used the horizontal stretch option, along with width of 100%:

width	100%
Layout (MatrixLayout)	
stretchedHorizontally	<input checked="" type="checkbox"/>
stretchedVertically	<input type="checkbox"/>


The groups themselves also have width set to 100%. By doing this my two columns cover the full width of the screen, rather than adjusting according to the lengths of the fields. In the groups I am not using either of the stretch options. So the fields and their labels are left-aligned, rather than spread out across the column:



## Assistance Class

[http://help.sap.com/saphelp\\_nw70/helpdata/en/43/bcd2b8e326332ee10000000a11466f/frameset.htm](http://help.sap.com/saphelp_nw70/helpdata/en/43/bcd2b8e326332ee10000000a11466f/frameset.htm)

You can create an assistance class for your webdynpro, which must inherit from CL\_WD\_COMPONENT\_ASSISTANCE. The name of the class just needs to be entered on the webdynpro component definition, underneath the description:

Web Dynpro Component	ZMSD_WBD_ORDER	Inactive
Description	Order details	
Assistance Class	ZCL_MSD_WBD_ORDER	

An instance of the class is created automatically for you at runtime. You can then access that instance from your methods using attribute WD\_ASSIST, which is provided automatically.

With an instance class in place, you are now able to use text symbols in your webdynpro, which are stored in the text pool of the assistance class.

Aside from texts, some judgement is required in deciding what to put in the assistance class, and what to put in the webdynpro itself. When deciding keep in mind:

- The assistance class may be used by any number of webdynpro components. Therefore an assistance class provides a means of re-using methods, attributes and constants between a number of different components. Originally 'faceless' components were designed to serve this purpose, but in practice assistance classes can achieve the same result more easily ( see [http://help.sap.com/saphelp\\_nw70/helpdata/en/e0/202d4108f08739e10000000a1550b0/frameset.htm](http://help.sap.com/saphelp_nw70/helpdata/en/e0/202d4108f08739e10000000a1550b0/frameset.htm) )
- The methods and attributes of the assistance class are available globally in the webdynpro component - you can access them from both component and view controllers.
- References to webdynpro context nodes and elements may be imported and used in the assistance class methods. But pass the references into the methods rather than storing them in the class - the context could change.

## Using the context

Coming from background of developing dynpro applications, I found it hard initially to understand how best to use the context. What follows are my thoughts on the subject so far - your suggestions, comments or corrections are most welcome.

First, it is helpful to be very clear about the various terms that are used in describing the context:

NODE - a data structure within the context.

ATTRIBUTE - a single data field under a node

ELEMENT - an instance of a node (there may be many, e.g. for a table of data)

Things to note about the context:

- Any data that you want to see in a view, must be in the context of the view controller.
- You can use context attributes to control UI elements dynamically (discussed in more detail below).
- You should not treat the context as a repository for all of your data.

This last point is a mistake I made myself, partly because of the way I used the Service Call wizard to call a BAPI (see below). It is more efficient, and easier to code, if you store and access your data through either controller attributes, or in your assistance class (I now use the assistance class).

Leaving aside attributes for controlling UI element properties, it may be helpful to think of the context in the same way as a screen structure in dynpro applications. Fill the context with current data before displaying the view. When responding to events, read the context data (which may have changed), and copy it back to your data structures. Any logic can then access the data structures directly, rather than having to repeatedly access the context via method calls.

- It may be useful to structure the context in the same way as your views.

For example you may intend to display data over a number of tabs. There may be some advantages in having a separate node defined for each tab.

- Context nodes are created with the 'singleton' property checked by default. You should normally switch this off - Thomas Szücs explains why:

<https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/4810>

Further advice on the context is available in SAP Help:

[http://help.sap.com/saphelp\\_nw04s/helpdata/en/8c/a0fa495f9a480bae29bf43474ccb79/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/8c/a0fa495f9a480bae29bf43474ccb79/content.htm)

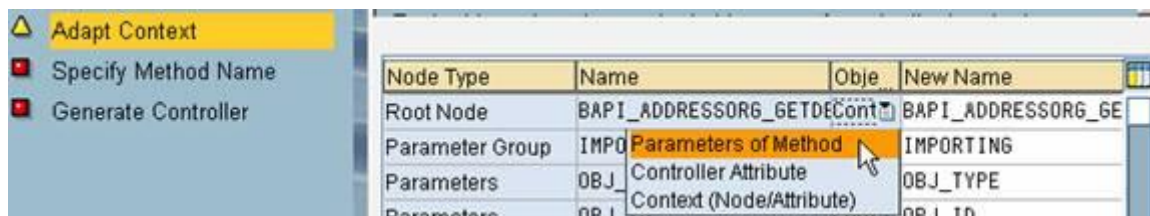
[http://help.sap.com/saphelp\\_nw04s/helpdata/en/7a/787e40417c6d1de10000000a1550b0/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/7a/787e40417c6d1de10000000a1550b0/content.htm)

### **The Service Call wizard and calling function modules**

You can use a wizard to create service calls to web services, function modules and class methods. The wizard will only create the calls in either the component controller or a custom controller. This fits with the MVC model - we should not be making service calls in the view controller, which should only be concerned with displaying data in the view. Note that using this Service Call option is not obligatory - it is quite possible to code service calls yourself.



In creating the service calls, the wizard can also create context nodes and elements for the interface parameters of the BAPI. The key thing to note is that you can choose how each parameter is implemented:



Rather unhelpfully, both 'Controller Attribute' and 'Context' are shown abbreviated as 'Cont'. So to be sure of what is selected, it may be useful to expand the width of this field so you can see the full description.

Do not make the mistake of creating everything in the Context:

- Many of the parameters won't need to be shown on views, or bound to UI attributes. Therefore they do not need to be in the context
- We may prefer to structure the context to reflect the way the views are structured, rather than taking the structure from the service interface.

With the other two options, either controller attributes or method parameters are created for you automatically. This may be useful, but doesn't save a huge amount of labour - personally I prefer to write my own function module calls, which leaves me full control over the process.

## Mandatory Fields

In webdynpro, UI input fields can be given status 'required'. This causes them to be marked on screen with a red star.



Unlike in dynpro applications, no check is made automatically to see if the fields have been filled. To make the check yourself, obviously one option is to code a simple check for each input field. However you may have an application that determines dynamically what input fields are mandatory - if so the check must be dynamic also. Fortunately a static method is available to perform such a check:

### **cl\_wd\_dynamic\_tool=>check\_mandatory\_attr\_on\_view**

- any fields with missing data produce a generic error message, and the field is highlighted.

## **Error Messages**

Messages are reported using methods of IF\_WD\_MESSAGE\_MANAGER.

- By using the REPORT\_ATTRIBUTE\_\* methods, you can link a message to a specific input field, which will be highlighted on the view.
- Unlike in dynpro programming, processing does not stop when you have raised an error message. Therefore you need to consider if you should exit the method after an error has been raised; or just continue, possibly to identify further errors to report.
- Similarly, raising an error message does not automatically cancel any screen navigation. If you want the user to stay on the same screen when an error message is raised, then you need to set the 'CANCEL\_NAVIGATION' parameter.

## **Responding to user input**


When creating SAPGUI-based dynpro applications, we can respond to user input in PAI. By using 'on input' and 'on request' we can react to changes to specific fields. It may be useful to do the same thing for a web dynpro view, but how can we know what has changed?

Fortunately, the **Context Change Log** is provided for this purpose:

[http://help.sap.com/saphelp\\_nw2004s/helpdata/en/ae/f95e42ff93c153e10000000a1550b0/content.htm](http://help.sap.com/saphelp_nw2004s/helpdata/en/ae/f95e42ff93c153e10000000a1550b0/content.htm)

Using methods of IF\_WD\_CONTEXT you can switch the log on and off, retrieve a table of changes or reset it. The table of changes tells you exactly what context attributes changed, along with the old and new values.

In dynpro applications, PAI runs following any kind of user action, such as pressing enter, or calling a search help, or pressing a button. In webdynpro we have control over when to respond to user changes. In the view layout, each UI element has a list of possible events as part of its properties. For example all input fields let you respond to the user hitting 'enter' on that field:

Events	
onEnter	SUBMIT 

In this example I've created an event called 'SUBMIT'. I've used this with all UI elements where I want to respond to the user's input. Within the event handler method, I read the context change log and respond to any changes that have been made.

### **Automatic refresh**

You may have an application where you want to automatically refresh the data selection periodically, without intervention from the user. This can be achieved using a TimedTrigger UI element:

[http://help.sap.com/saphelp\\_nw04s/helpdata/en/da/a6884121a41c09e10000000a155106/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/da/a6884121a41c09e10000000a155106/content.htm)

### **Drill down**

Within SAPGUI applications it's common to provide a drill down facility from an ALV grid. For example if you are reporting a list of order numbers, you may want the ability to click on an order number and have the order appear. This kind of facility is not so easy to provide from webdynpro - unless you are drilling down to a another webdynpro. The following SDN thread discusses the options:

<https://www.sdn.sap.com/irj/sdn/thread?threadID=269990>

The favoured solution seems to be creating a shortcut object. When called this opens SAPGUI, navigates to the transaction required, and can also fill parameter fields on the front screen. Skipping directly to the next screen does not seem to be possible, unlike when using a call transaction. Function SWN\_CREATE\_SHORTCUT may be helpful for creating the shortcut.

### **Dynamically changing UI Elements**

Something you may want to do in response to user input, is to dynamically change properties of your UI elements. Specifically, we may want to change:

- Visibility
- Read Only
- Required
- Enabled

There are at least three ways of doing this:

#### **1) Binding to Context Attributes**

In this method you create context attributes and bind them to the properties of your UI elements. This may be particularly effective when a large number of UI elements should behave in the same way for particular situation. For example I may have an application that can run in 'change' or 'display' modes. I could create a single 'readonly' context attribute, and bind it to the 'readOnly' property of all

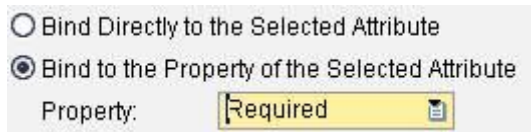
my input fields. When the user switches between display and change modes, I can set the value of my context attribute to switch the field properties.

## 2) Binding to Context Attribute Properties

For a more complex application you may want to control the four attributes of every input field individually. Using the above method, this would involve creating huge numbers of context elements (four for every input field). Fortunately this is not necessary - context attributes have these four properties available automatically:

[http://help.sap.com/saphelp\\_nw04s/helpdata/en/45/d44e0cdf073b1e1000000a11466f/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/45/d44e0cdf073b1e1000000a11466f/content.htm)

To use the context attribute properties: when binding select the context attribute and then choose 'Bind to the Property of the Selected Attribute', along with the actual property from the dropdown.



The image shows a configuration dialog with two radio button options and a dropdown menu. The first option is 'Bind Directly to the Selected Attribute' with an unselected radio button. The second option is 'Bind to the Property of the Selected Attribute' with a selected radio button. Below the options is a label 'Property:' followed by a dropdown menu showing the value 'Required' and a small icon to the right.

## 3) Dynamic Programming using WDDOMODIFYVIEW

For each view, inside hook method WDDOMODIFYVIEW you are given a reference to the view itself. Using this it is possible to get references to the view's UI elements. Using methods of the various UI element classes, you can change the UI element properties directly, without them being bound to anything in the context.

Note that WDDOMODIFYVIEW is the only place you can get the view reference, and should be the only place where you make these kinds of changes.

Because the UI elements are not bound, this method has the advantage that you can still set the properties in the view layout, which will be your default values. You only need the dynamic programming to run when you want to change from those defaults.

Each of these three methods has its own advantages and disadvantages, and the best method to use may vary according to circumstances. Something you may wish to consider is the use of a bespoke configuration table to control the UI properties. Such a table will need these fields:

- The data fields which you want to affect the layout. For example these could be Document Type, Company Code, application mode etc. These will be key fields.
- Fields to identify the UI element. These will probably be Webdynpro Component, View name and Element name. These will also need to be key.
- Simple flags for the four UI properties.

- A free text comment field may be useful for explaining your logic in words .

In my own application I used a config table like this, combined with dynamic programming. The UI elements took their default properties from the view layout, and my table only needed to control when they changed from those defaults. I coded my selection so that I could create generic config records, such as a rule to apply to all fields in display mode. That's to say, you don't need a config record for every possible combination of key field values. To make this possible I just had to allow space as a valid value in my select:

```
select *
  from zmsd_wbd_fields
 into table wd_this->gt_field_config
 where auart      in (space, x_auart)
    and vbtyp     in (space, x_vbtyp)
    and poart     in (space, x_poart)
    and appl_mode in (space, x_appl_mode)
```

Note that you could bind to context attribute properties (method 2 above), and still use a configuration table in this way. You would however need more records, as you could not define default values on the view layout.

### **Calling Input Help dynamically**

The various methods of implementing input help are described in SAP Help:

[http://help.sap.com/saphelp\\_nw04s/helpdata/en/9b/c51c42735b5133e10000000a155106/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/9b/c51c42735b5133e10000000a155106/content.htm)

But what if you want to call input help dynamically from your method code? For example you may want the input help to appear automatically, in response to the user changing an input field. A suitable method is available, but it's SAP internal:

#### **cl\_wdr\_value\_help\_handler=>handle\_value\_help**

To use it we simply need to pass in a reference to the context element, and the name of an attribute. It will then run the value help defined in the context for that attribute.

However being SAP internal, the method could change or disappear at any time. So if anyone can find an alternative way to trigger a help dialog, then please let us know and I'll update this section.

### **Application Parameters**

When creating Webdynpro applications, various parameters can be set that alter the behaviour of the application:

Application: zmsd_wbd_order_create Saved		
Parameters		
Parameters	Value	Description
WDFORCEEXTERNALSTYLESHEET	X	Force Use of External Stylesheet
WDELTA_RENDERING	ON	Delta Rendering

[http://help.sap.com/saphelp\\_nw70/helpdata/en/7b/fb57412df8091de10000000a155106/content.htm](http://help.sap.com/saphelp_nw70/helpdata/en/7b/fb57412df8091de10000000a155106/content.htm)

Some of these options affect the look and feel of the application and are described below.

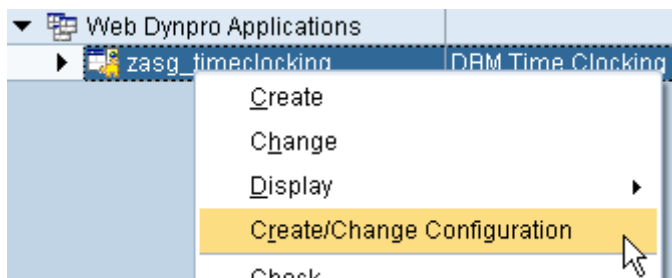
Additional parameters that may be of particular interest are:

- WDELTA\_RENDERING (performance)
- WDSHAREDREPOSITORY (performance)
- WDPROTECTEDAPPLICATION (security)

### Configuration

Configuration offers a further option for changing view layouts. Configuration may be thought of as being equivalent to transaction variants in dynpro applications. Within a configuration, the properties of UI elements may be changed in much the same way as the dynamic options described above.

[http://help.sap.com/saphelp\\_nw70/helpdata/EN/22/719f42f2ff7e5fe10000000a155106/content.htm](http://help.sap.com/saphelp_nw70/helpdata/EN/22/719f42f2ff7e5fe10000000a155106/content.htm)



Once created, a configuration is linked to an application using parameter WDCONFIGURATIONID. This makes it possible to create a number of different applications, which are all just versions of the same underlying component.

To the best of my knowledge, a configuration cannot be changed dynamically at runtime. So for example if you created separate configurations for 'Display' and 'Change' modes, the user would not be able to switch between them at runtime. Therefore in that situation, the methods to change UI elements dynamically may be more useful.

### Look and Feel

The look and feel of your ABAP webdynpro applications can be changed using themes. A theme is a collection of style sheets (.CSS files) and graphics.

This ability is highly significant: your applications are *not* restricted to the standard SAP look and feel. Minor changes such as text size can easily be made via the theme editor (described below). With more extensive CSS knowledge, the entire look and feel of the application can be customised. An obvious use for this would be to apply corporate branding.

When running the webdynpro through a **portal**, application parameter WDFORCEEXTERNALSTYLESHEET may be useful to ensure that webdynpro always uses the style sheet from the portal. Themes may be developed using the Theme Editor supplied with the portal:

[http://help.sap.com/erp2005\\_ehp\\_03/helpdata/EN/95/8bfe40f652f323e1000000a155106/content.htm](http://help.sap.com/erp2005_ehp_03/helpdata/EN/95/8bfe40f652f323e1000000a155106/content.htm)

Alternatively, application parameter WDTHEMEROOT may be used to specify a theme for the webdynpro, different to that of the portal. A number of standard themes are available to choose from. The exact logic for theme determination is explained in SAP help, and also on the SDN:

[http://help.sap.com/saphelp\\_nw04s/helpdata/en/46/89af7fbe4d429ee1000000a1553f7/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/46/89af7fbe4d429ee1000000a1553f7/content.htm)  
<https://www.sdn.sap.com/irj/sdn/thread?threadID=511592>

Without a portal, we may also like to create and use bespoke themes. The Theme Editor supplied with the portal, can also be downloaded from the SDN to be used stand-alone:

<https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/136dd890-0201-0010-1b9d-bd09a0d3b1d8>

This is quite a complex procedure, requiring you to install Java and Eclipse, and then adding a plug-in to Eclipse for the theme editor. Once complete you can copy a standard theme, such as sap\_tradeshaw, and then modify it.

Be careful to download the versions of Java and Eclipse specified - it may not work with more recent versions. If you have more than one version of the Java runtime environment installed, you may need an extra '-vm' parameter to make it work - explained here:

<https://www.sdn.sap.com/irj/sdn/thread?messageID=5421135#5421135>

In my own case I had difficulty in adding the parameters to my Eclipse shortcut, so instead I added to .cmd file to my desktop with the following script:

```
C:  
cd \  
cd Program Files\Eclipse  
eclipse.exe -vm "C:\Program Files\Java\j2re1.4.2_17\bin\javaw.exe" -consolelog -vmargs -Xmx512M
```

Once we have created our own theme, we need to upload it to the MIME repository and then use it in our webdynpro application. This process is explained here, by Bastian Preissler and Matthias Grün:

<https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/7015b1f9-535c-2910-c8b7-e681fe75aaf8>

Two possible URL parameters are mentioned here: I found 'SAP-EP-THEMEROOT' to work much better than 'SAP-CSSURL'.

Note that it is possible to set the theme from within the webdynpro code. This is enormously powerful: if required, your application could run logic to choose from a number of themes at runtime.

Here's a screenshot from an application using a modified theme. The text is all much larger than normal, and the colours used in the table have been changed:

◀ Back **Romford - Test Terminal Roger**

Michael Alosso

Order Number:  Enter ▶

▶ Keyboard / List

Order	Customer Name	License Plate
2000003198	Jim Carey	A994TEV
2000003204	Penelope Parker (BP)	MSLVW01
2000003210	Jim Carey	A994TEV

### Running webdynpro applications via SAP GUI

It is very simple to create a SAP GUI transaction that will run a webdynpro application.

The transaction can be created in SE93, calling transaction WDYID with the application name as one of the parameters (thanks to Glen Simpson for this tip):

[http://help.sap.com/saphelp\\_nw04s/helpdata/en/43/27ca50d51e0b19e10000000a1553f6/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/43/27ca50d51e0b19e10000000a1553f6/content.htm)

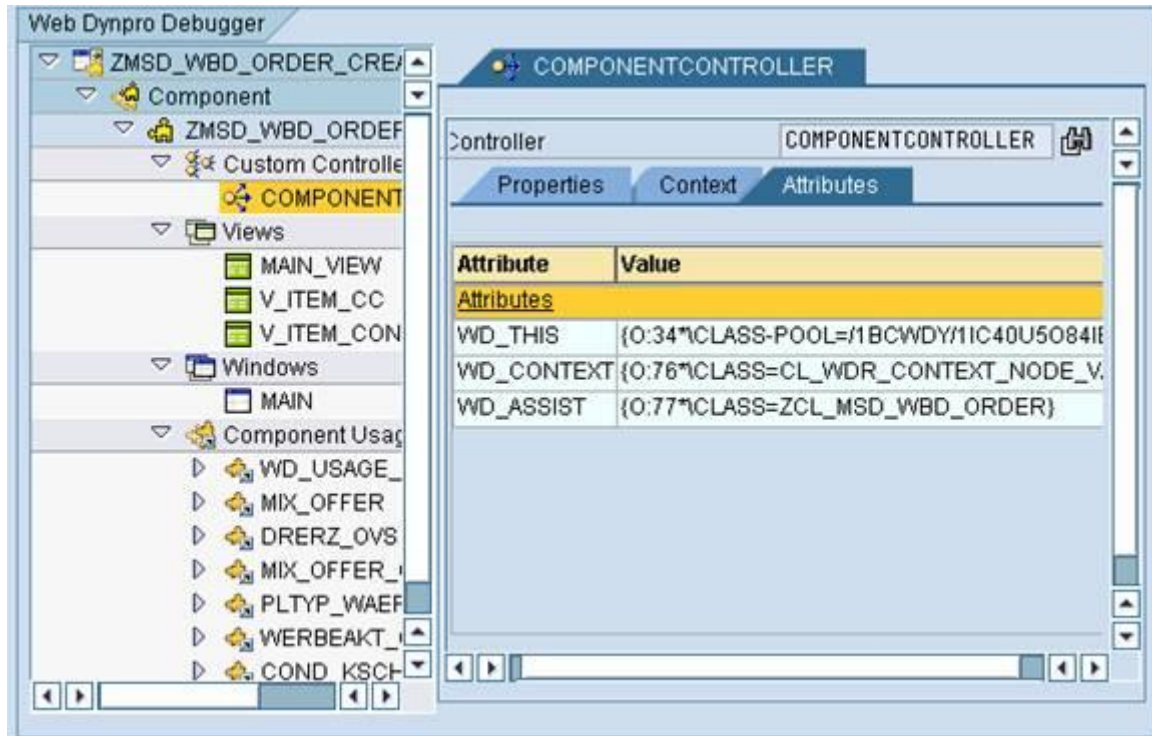
Transaction WDYID uses function module WDY\_EXECUTE\_IN\_PLACE. This may be used to run a webdynpro application inside a browser control on a dynpro screen.

[http://help.sap.com/saphelp\\_erp2005/helpdata/en/43/2f1a63cb883575e10000000a11466f/content.htm](http://help.sap.com/saphelp_erp2005/helpdata/en/43/2f1a63cb883575e10000000a11466f/content.htm)

## Debugging

Webdynpro ABAP code can be debugged by setting external breakpoints. When the breakpoints are hit, the (new) debugger will be launched. Note that you must not already have the maximum number of SAP GUI sessions open - if you do the debugger cannot launch and the application will not stop.

Within the debugger a special tool is available to help you to debug the webdynpro itself:



[http://help.sap.com/saphelp\\_nw04s/helpdata/en/43/95a6c84f7c60b7e10000000a11466f/content.htm](http://help.sap.com/saphelp_nw04s/helpdata/en/43/95a6c84f7c60b7e10000000a11466f/content.htm)

## Example webdynpro application

An example of a full-scale, complex, webdynpro is **LORD\_MAINTAIN\_COMP**. This can be used to create, maintain and display sales orders.

## The Future: forthcoming features

ABAP Webdynpro is still evolving, and Netweaver enhancement pack 1 (NW 7.01) brings a selection of excellent new features. Some of the most eye-catching are:

- **'Light Speed' rendering.** This is a massive leap forward - quite simply web dynpros will have much better performance and a more 'web 2.0' look and feel. Expect a look and feel remarkably similar to CRM 2007.
- **Drag & Drop:** until now, it has not been possible to put drag and drop functionality into your webdynpros - with EhP 1 this becomes possible. For NW 7 and below, check out David Lees' excellent

blog describing drag and drop using dynpro controls

technology:<https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/12839>

- **Flash Islands:** it is now possible to integrate Adobe Flash into your webdynpros.

I have yet to develop on a 7.01 system, so I can't offer any further insight on the features just yet.

However much more detailed information is available in SAP Help:

[http://help.sap.com/saphelp\\_nw70ehp1/helpdata/en/48/01661984ff4aa5e10000000a421937/frameset.htm](http://help.sap.com/saphelp_nw70ehp1/helpdata/en/48/01661984ff4aa5e10000000a421937/frameset.htm)

**Source:** <http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/8402>