Enhancing Web Dynpro Table Performance

Bertram Ganz, Andreas Rössler, NW ESI Foundation - Web Dynpro Foundation for Java

Overview

The existing Web Dynpro table selection behavior is automatically combined with a corresponding lead selection change. When clicking on a table cell editor, it is even possible that two separate server roundtrips have to be processed, which may lead to an undesired performance drawback.

In this article, you will learn how to improve the Web Dynpro table selection behavior using the newly introduced (NetWeaver 04 SP Stack 13) table property *compatibilityMode*. Combined with the implementation of a parameter mapping relation, table performance can be enhanced significantly.

Content

Keywords

Tables, table performance, table selection behavior, table cell editor, implicit lead selection change, compatibilityMode, readOnly, parameter mapping, NetWeaver 04 SP Stack 13

1. Introduction

1.1 Table Selection Behavior Before NW04 SP Stack 13

The Web Dynpro event model is based on an event abstraction known as *action* to cause client-side event information to be transported to the server. Actions are the connection between an event fired by a UI element and the corresponding event handling code implemented as an *action event handler* in a Web Dynpro view controller.

The Web Dynpro programming model generally allows you to handle only a single action (user interaction) per server roundtrip; but in one special case this paradigm is not fulfilled: *table cell editor interaction*. When a user triggers an action inside a cell editor that does not belong to the selected table line, two server roundtrips occur.

The Web Dynpro *Table UI element* is able to nest *UI elements* as *table cell editors*. Some of them have events for firing their own actions to be handled by the application (like *Button*, *LinkToAction*, *Checkbox*, and so on). During runtime, this *TableCellEditor* will be rendered for each node element of the table's *collection* that corresponds to the table lines. Therefore, there is just one *UI element* in the UI tree but several visualized instances on the rendered screen content. If an application uses, for example, a button in a Web Dynpro table and the user presses one of the visualized instances, the application has to determine which instance was pressed.

In NetWeaver 04, the *lead selection* of the table is therefore implicitly set to the row element of the pressed button, before processing the action of the button itself. Now the application can get the row information of the pressed button by simply accessing the current context node element (*lead selection)*.

As a result, <u>two</u> actions may have to be processed:

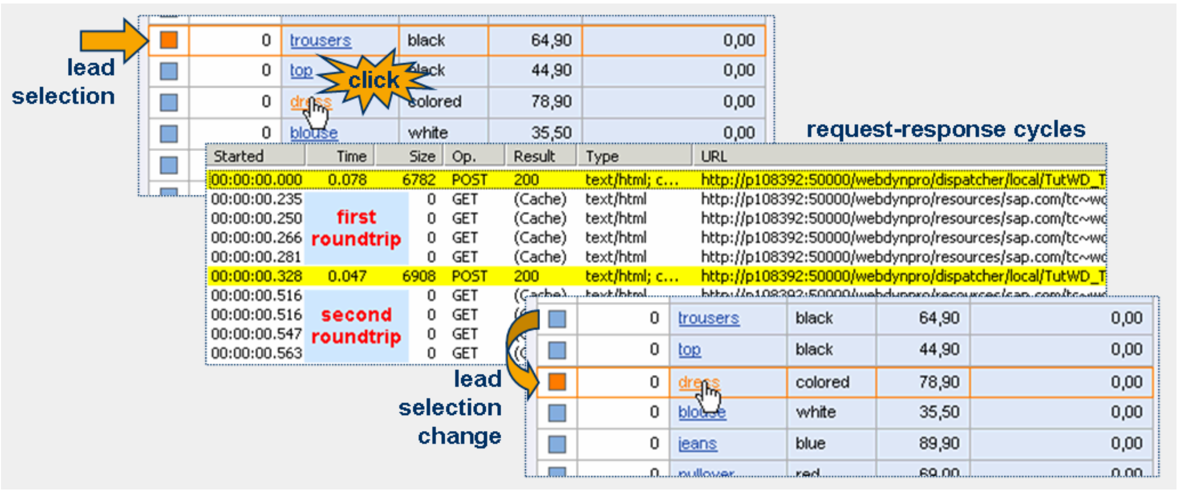
1. Implicit *leadSelectionChange* action of the table
2. Explicit action of the *TableCellEditor* itself

This can lead to severe problems, especially if one action disables the other – for example, the first action could navigate the screen to another view assembly and the *UI element* for triggering the second action would no longer be accessible.

To achieve a uniform table behavior, the *leadSelection* is now changed whenever the user interacts with a table cell that is not already *leadSelected*. Unfortunately, this implicit lead selection change can lead to many server roundtrips and longer response times. Due to compatibility reasons, this behavior must still be supported.

Example

Figure 1 demonstrates that two server roundtrips (POST requests) occur when the user clicks on a *LinkToAction* cell editor in another table line. In the first roundtrip, the lead selection is implicitly moved to another node element (table line); the second one handles the action event. The total request/response time is nearly twice as long as the second roundtrip.



🟧	0	trousers	black	64,90	0,00
🟦	0	top	black	44,90	0,00
🟦	0	dress	colored	78,90	0,00
🟦	0	blouse	white	35,50	0,00

Started	Time	Size	Op.	Result	Type	URL
00:00:00.000	0.078	6782	POST	200	text/html; c...	http://p108392:50000/webdynpro/dispatcher/local/TutWD_T
00:00:00.235		0	GET	(Cache)	text/html	http://p108392:50000/webdynpro/resources/sap.com/tc~wd
00:00:00.250		0	GET	(Cache)	text/html	http://p108392:50000/webdynpro/resources/sap.com/tc~wd
00:00:00.266		0	GET	(Cache)	text/html	http://p108392:50000/webdynpro/resources/sap.com/tc~wd
00:00:00.281		0	GET	(Cache)	text/html	http://p108392:50000/webdynpro/resources/sap.com/tc~wd
00:00:00.328	0.047	6908	POST	200	text/html; c...	http://p108392:50000/webdynpro/dispatcher/local/TutWD_T
00:00:00.516		0	GET	(Cache)	text/html	http://p108392:50000/webdynpro/resources/sap.com/tc~wd
00:00:00.516		0	GET			
00:00:00.547		0	GET			
00:00:00.563		0	GET			

🟦	0	trousers	black	64,90	0,00
🟦	0	top	black	44,90	0,00
🟧	0	dress	colored	78,90	0,00
🟦	0	blouse	white	35,50	0,00
🟦	0	jeans	blue	89,90	0,00
🟦	0	pullover	red	69,00	0,00

Figure 1: Two server roundtrips occur when a user triggers another cell editor action

1.2 New Table Selection Behavior in NW04 SP Stack 13

In NW04 SP Stack 13, a new table selection behavior was introduced in order to enhance the table interaction performance. With this new feature, the application developer can eliminate the second implicit server roundtrip (for moving the lead selection) when a user clicks a table cell editor that is bound to an action.

In addition, the new behavior allows developers to edit several input fields in separate table lines, without repeatedly changing the lead selection. When editing several fields in a table, this new selection behavior improves table performance.

Using the New Table Property compatibilityMode

The solution is based on introducing a new and universal event parameter *nodeElement* to all UI element events. With this new parameter, the Web Dynpro client can pass a node element reference to the view controller with a single action. In other words, the Web Dynpro client can pass a node element reference related to the clicked table cell editor (in a certain table row) without moving the lead selection.

For enhancing the table interaction performance based on this new behavior the new table property *compatibilityMode* must be defined. Additionally, the application developer must implement some controller code for mapping the new UI element parameter *nodeElement* to a parameter of the action event handler.

Besides the omission of a second server roundtrip, the new table selection behavior differs from the former with regard to the implicit lead selection change:

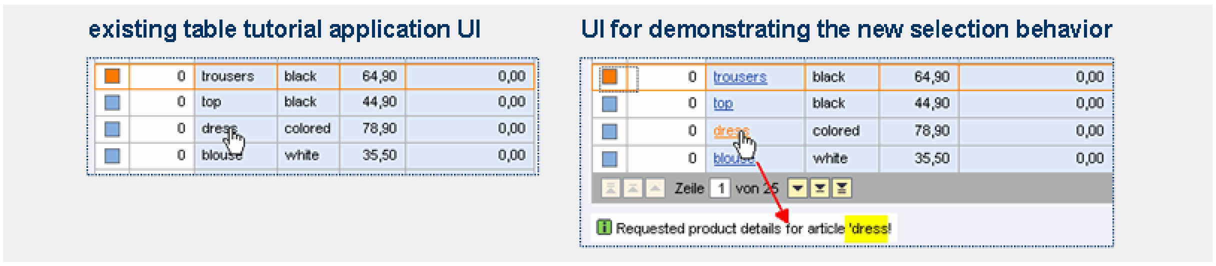
- **`Table.readOnly=false:`** The lead selection is only changed by using the selection column (first column) of the table.
- **`Table.readOnly==true:`** The lead selection is changed whenever the user interacts with a table cell that is not already lead-selected and is not bound to an action. If a cell editor is bound to an action, the selection is *not* changed. This means that if the user presses a button in a table, only the button action is triggered but the lead selection remains unaffected.

2. Enhancing Table Selection Performance

To enhance table selection performance based on the new (NW 04 SP Stack 13) table property *compatibilityMode,* the application developer must explicitly implement a corresponding *parameter mapping* relation. This means that this enhancement cannot be achieved by purely declarative means.

2.1 Example: Showing Product Details by Clicking an Article Name

To demonstrate the new table selection behavior, we will extend the existing Web Dynpro table tutorial application. The article column contains a *LinkToAction* cell editor instead of a *TextView.* To display the details for a specific product the user can click every article name without implicitly changing the table's lead selection:



existing table tutorial application UI

🟧	0	trousers	black	64,90	0,00
🟦	0	top	black	44,90	0,00
🟦	0	dress	colored	78,90	0,00
🟦	0	blouse	white	35,50	0,00

UI for demonstrating the new selection behavior

🟧	0	trousers	black	64,90	0,00
🟦	0	top	black	44,90	0,00
🟦	0	dress	colored	78,90	0,00
🟦	0	blouse	white	35,50	0,00

Requested product details for article 'dress!

2.2 Parameter Mapping

The former selection behavior used the *lead selection* information to determine the table line (node element) in which the user clicked a certain cell editor. Consequently, the corresponding context node element can simply be accessed by calling `wdContext.current<node name>Element()`. Without an implicit lead selection change we can no longer access the clicked node element (the node element to which the clicked cell editor is related) by calling the method `current<node name>Element()`.

Instead, a *parameter mapping relation* must be implemented for transferring the UI element event parameter *nodeElement* to another parameter in the related action event handler (see figure 2). With this mechanism a reference to the clicked node element (product in table line) is automatically passed to the action event handler by the Web Dynpro runtime. The lead selection no longer needs to be implicitly changed for this purpose.

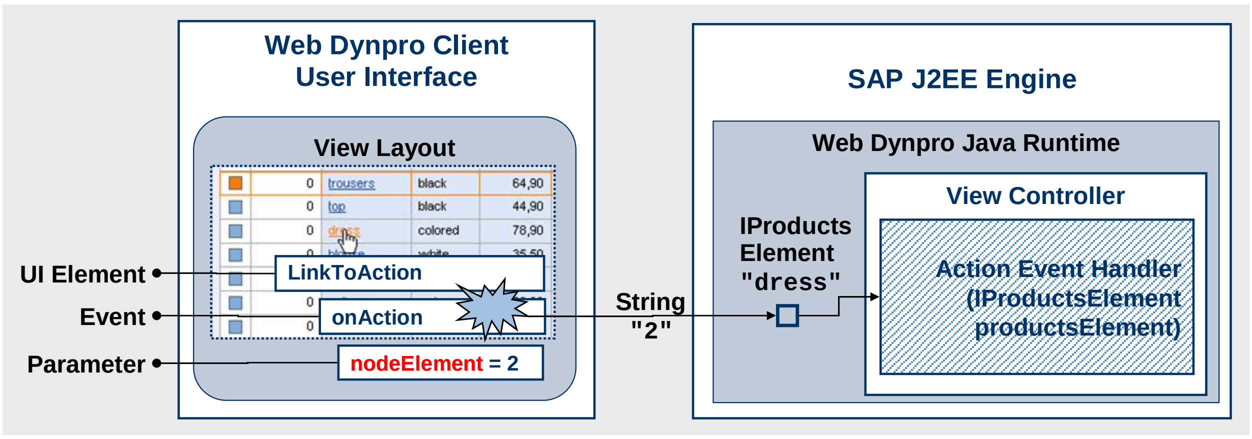


Figure 2: Parameter mapping between UI element event parameter nodeElement and action event handler

Setting the Table Property compatibilityMode=nw04Plus

In a first declarative step, we set the new table property *compatibilityMode* as **`nw04Plus`**. The default property **`auto`** represents the former table selection behavior. With the new *compatibilityMode* property the table can switch between the former and the new table selection behavior.

Adding the New Parameter productsElement to the Action Event Handler

In our second step we:

- Define a new action *ShowDetails.*
- Define a new parameter for the new action ShowDetails.
    - *parameter name*: since the table is bound to the context node *Products*, the new parameter is named *productsElement*.
    - *parameter type*: the parameter type must be defined as `IPrivate<view name>.I<node name>Element`, in our case `IPrivateCompatibilityModeView.IProductsElement`. If no typed access to the context exists, the parameter type must be `IWDNodeElement`.
- Bind the *onAction* event of the *LinkToAction* table cell editor to the new action *ShowDetails*.

After this step, the related action event handler *onActionShowDetails()* automatically contains the correctly typed parameter *productsElement*.

Implementing the Parameter Mapping

The controller code, which implements the required parameter mapping, is as follows:

View Controller TableCompatibilityModeView.java

```
public static void wdDoModifyView(
  IPrivateTableCompatibilityModeView wdThis,
  IPrivateTableCompatibilityModeView.IContextNode wdContext,
  com.sap.tc.webdynpro.progmodel.api.IWDView view,
  boolean firstTime)
{
  //@@begin wdDoModifyView
  if (firstTime) {
  ...
  // Apply compatibilityMode feature for omitting a second server roundtrip:
  // Map UI element event parameter 'nodeElement' to action (event handler)
  // parameter 'productsElement' (parameter mapping).
  IWDLinkToAction linkToAction =
    (IWDLinkToAction) View.getElement("ARTICLE_editor");
  linkToAction.mappingOfOnAction()
    .addSourceMapping("nodeElement", "productsElement");
  }
  //@@end
}
```

Controller code (1): Implementing the parameter mapping

2.2 Accessing the Clicked Node Element

We can now easily access the clicked node element within the action event handler *onActionShowDetails()*using the passed action parameter *productsElement* of the type `IPrivateCompatibilityModeView.IProductsElement`. The context node's lead selection does not have to be changed for this purpose.

View Controller TableCompatibilityModeView.java

```
//@@begin javadoc:onActionShowDetails(ServerEvent)
/** Declared validating event handler. */
//@@end
public void onActionShowDetails(
  com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent,
  com.sap.tut.wd.tutwd_table.tablecomp.wdp
    .IPrivateTableCompatibilityModeView.IProductsElement productsElement )
{
  //@@begin onActionShowDetails(ServerEvent)

  // access node element instance related to the selected table line with the 
retrieved
  // event parameter. The lead selection must not be the same.
  String articleName = productsElement.getARTICLE();
  wdComponentAPI.getMessageManager().reportMessage(
    IMessageTableComp.SHOW_PRODUCT_DETAILS,new Object[] {articleName}, true);
  //@@end
}
```

Controller code (2): Accessing the clicked node element within the action event handler

3. Comparison of Former and New Table Selection Behavior

Figure 3 helps us to better understand the differences between the former and the new table selection behavior.

1. The table selection behavior depends on the two properties *compatibilityMode* and *readOnly*. The new selection behavior now also depends on the table's *readOnly* value.
2. The table selection behavior depends on different click locations.
3. The table selection behavior is associated with a different number of roundtrips as well as an implicit, explicit, or omitted lead selection change.

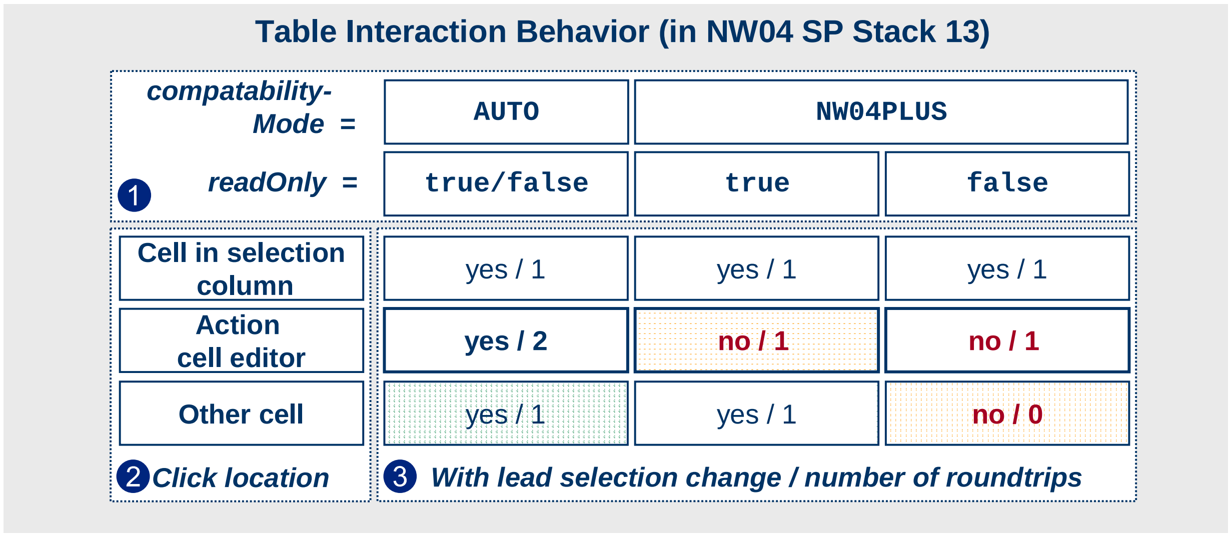


Table Interaction Behavior (in NW04 SP Stack 13)

compatability-Mode =	AUTO	NW04PLUS	
1 *readOnly* =	true/false	true	false
Cell in selection column	yes / 1	yes / 1	yes / 1
Action cell editor	**yes / 2**	**no / 1**	**no / 1**
Other cell	yes / 1	yes / 1	**no / 0**
2 *Click location*	3 *With lead selection change / number of roundtrips*		

Figure 3: Comparison of different table interaction behaviors (lead selection change, roundtrips)

As you can see, the new table selection behavior eliminates the second server roundtrip when triggering an action by clicking a table cell editor. In this case, the lead selection no longer changes implicitly.

In the case of editable tables (*readOnly* = *false*), the lead selection no longer changes when another table cell is clicked.

4. Sample Application

This article is based on a Web Dynpro sample application, which is available for download on the *SAP Developer Network (SDN)* at http://sdn.sap.com under *Home → Developer Areas → Web Application Server → Web Dynpro → Code Samples → Sample Applications and Tutorials → Working with Tables in Web Dynpro (11) → B) Enhancing Table Performance* (Permalink).

Prerequisites

- The sample application is based on a *SAP NetWeaver 04 – Stack 13* installation.

Importing the Project

1. Call the *SAP NetWeaver Developer Network (SDN)* via the URL http://sdn.sap.com and log on with your user ID and password. If you do not have a user ID, you must register before you can proceed.
2. Move to *Home → Developer Areas → Web Application Server → Web Dynpro → Code Samples → Sample Applications and Tutorials → Working with Tables in Web Dynpro (11) → B) Enhancing Table Performance* (Permalink).
3. Download the zip file containing the Web Dynpro project *WD_Table_CompatibilityMode* and save the zip file to any directory on your local hard disk or directly in the work area of the SAP NetWeaver Developer Studio.
4. Extract the content of the zip file to the work area of the SAP NetWeaver Developer Studio or any directory on your local hard disk.
5. Open the SAP NetWeaver Developer Studio.
6. Import the Web Dynpro project *TutWD_Table_CompatibilityMode*:
    a. In the menu, choose *File → Import*.
    b. In the next window, choose *Existing Project into Workspace* and choose *Next* to confirm.
    c. Choose *Browse*, open the folder in which you saved the project *TutWD_ Table_CompatibilityMode*, and select the project.
    d. Choose *Finish* to confirm.

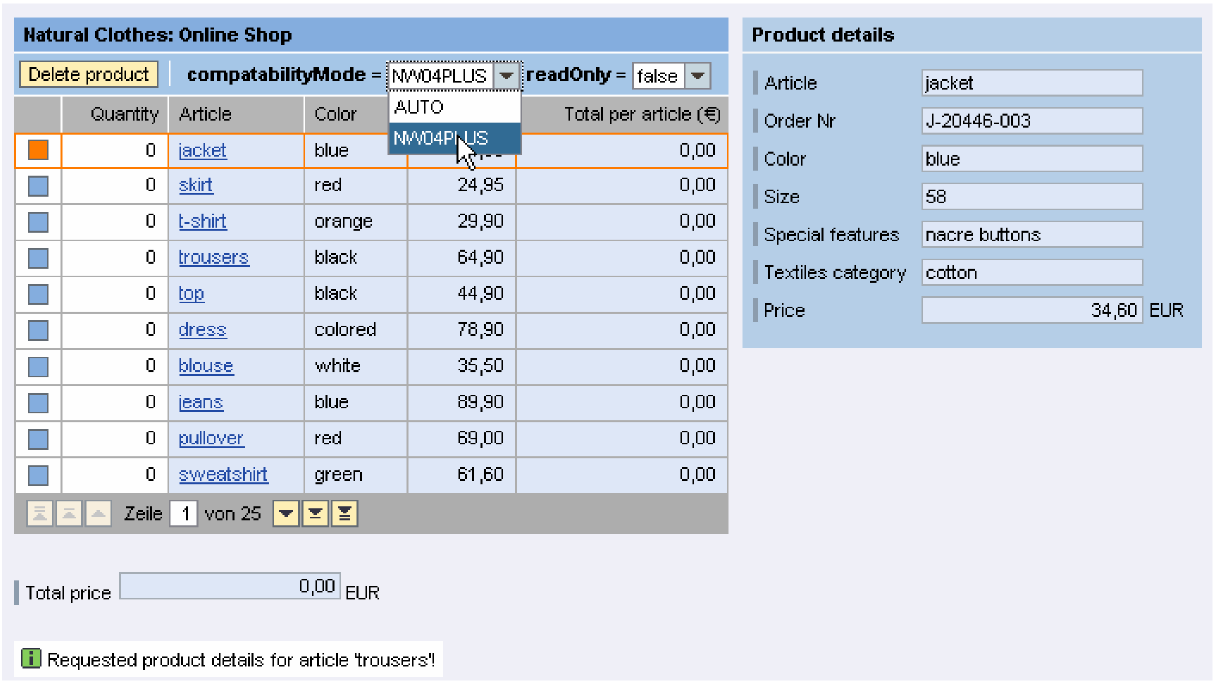
The Web Dynpro project *TutWD_Table_CompatibilityMode* appears in the Web Dynpro Explorer.

7. In the Web Dynpro Explorer, open the node *TutWD_Table_CompatibilityMode* → *Applications* → *TableCompatibilityModeApp.*
8. In the node's context menu, choose *Deploy new Archive and Run*. The sample application is then started in a new browser window.

Testing the Different Table Selection Change Behaviors

Depending on the specific settings of the two table properties *compatibilityMode* and *readOnly*, you can observe a different table selection change behavior.

Try to reproduce the different behaviors (as listed in Figure 3) interactively by changing the two table properties *compatibilityMode* and *readOnly* with the two dropdown lists.



Natural Clothes: Online Shop

Delete product **compatabilityMode** = NW04PLUS **readOnly** = false

	Quantity	Article	Color		Total per article (€)
🟧	0	jacket	blue	[illegible]	0,00
🟦	0	skirt	red	24,95	0,00
🟦	0	t-shirt	orange	29,90	0,00
🟦	0	trousers	black	64,90	0,00
🟦	0	top	black	44,90	0,00
🟦	0	dress	colored	78,90	0,00
🟦	0	blouse	white	35,50	0,00
🟦	0	jeans	blue	89,90	0,00
🟦	0	pullover	red	69,00	0,00
🟦	0	sweatshirt	green	61,60	0,00

Zeile 1 von 25

Product details

Article	jacket
Order Nr	J-20446-003
Color	blue
Size	58
Special features	nacre buttons
Textiles category	cotton
Price	34,60 EUR

Total price 0,00 EUR

Requested product details for article 'trousers'!