

DEVELOPING STATE-OF-THE-ART TABLE USER INTERFACES IN WEB DYNPRO JAVA

CD254

Exercises / Solutions

BERTRAM GANZ
SAP NETWEAVER UI F OPS, SAP AG


PETER BARKER
NETWEAVER UI PRODUCT MANAGEMENT, SAP AG

CHRIS WHEALY
NETWEAVER RIG, SAP (UK) LTD

Exercise 1 – Using Table Cell Variants

This first exercise dealing with Web Dynpro Tables in SAP NetWeaver 7.0 is based on an incomplete Web Dynpro Project template. After having finished this exercise the following learning objectives will be fulfilled:

- Defining a calculated context attribute returning table cell variant keys for node elements (table rows).
- Adding a table cell variant of type *TableStandardCell* to a *TableColumn* UI element
- Adding a cell editor to a table cell variant.
- Binding a *TableColumn* UI Element to a calculated context attribute returning variant keys.
- Implementing a calculated context attribute getter method returning cell variant keys for table node elements.

 Estimated time required for completion: 20 min

Related Web Dynpro Project and View Layout

This Exercise is based on the following Web Dynpro project and view layout:

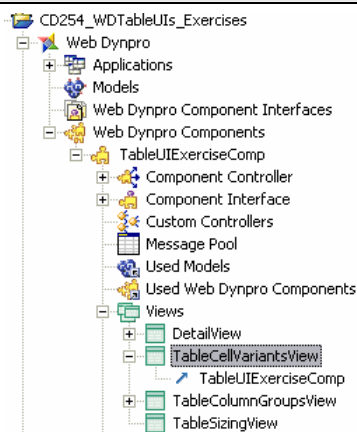
- **CD254_TableUIs_Exercises:** Web Dynpro project template comprising the initial state of the Web Dynpro views this exercises are based on.
- **TableCellVariantsView:** Exercise view for adding a cell variant to a table column.

The exercise solution can be found in the separate Web Dynpro project **CD254_TableUIs_Final**, view **TableCellVariantsView**.

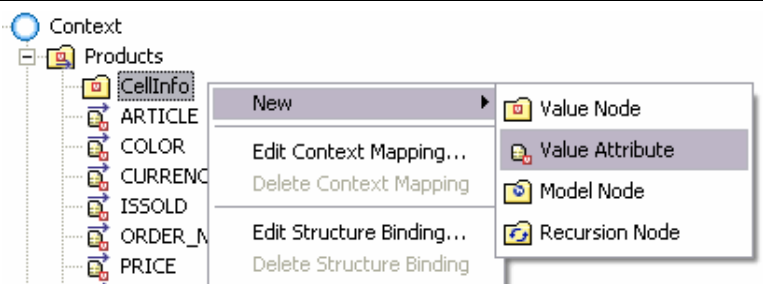
STEP 1 – Defining Required Calculated Context Attribute Returning Cell Variant Key

Every table node element storing table data must be enriched with a context attribute storing the corresponding selected cell variant ID. This is done by first adding a non-singleton node named *CellInfo* with cardinality 1..1 to the mapped table data node in the view controller context and then adding a calculated context attribute of type *String* to return the required *variantKey* per node element at runtime.

1. Open the Web Dynpro project **CD254_TableUIs_Exercises** within the Web Dynpro Explorer.
2. Navigate to tree node *Web Dynpro – Web Dynpro Components – TableUIExerciseComponent – Views – TableCellVariantsView*



3. Select tab *Context* to extend the pre-defined context structure with additional elements.
4. Open context menu for non-singleton child node **Products.CellInfo** and select menu item *New* → *Value Attribute*
5. Enter name **QUANTITY_SelCellVariantKeyCalc** and press button *Finish*.





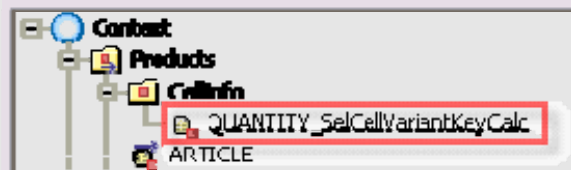
- In the *Properties* tab set property *calculated* to **true**. Also set property *readOnly* to **true**.

Properties	
Property	Value
calculated	true
calculatedAttributeGetter	getCellInfoQUANTITY_SelCellVariantKeyCalc
name	QUANTITY_SelCellVariantKeyCalc
readOnly	true
structureElement	
type	string

Result

The new calculated context attribute **QUANTITY_SelCellVariantKeyCalc** is added to the view context. The related context attribute getter method returns the table cell variant key for every displayed table row i.e. table node element.

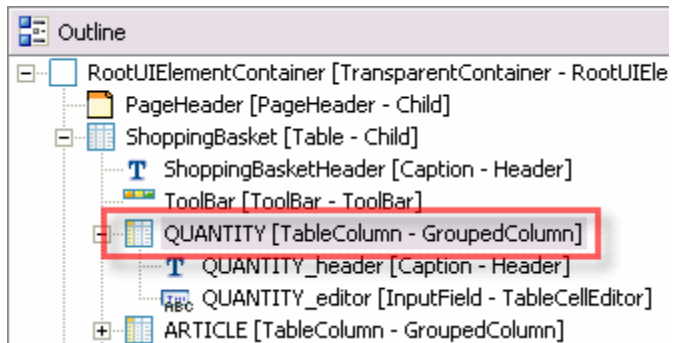
Context



STEP 2 – Adding a Cell Variant to a TableColumn UI Element

Next we add a cell variant of type *TableStandardCell* to the first table column *QUANTITY*. In this cell variant we add a table cell editor of type *TextView*. Finally, the new cell variant is specified with a corresponding variant key.

- Select tab *Layout* of view *TableCellVariantsView* to add a new cell variant to the first table column.
- In the *Outline* view select *TableColumn* UI element **RootUIElementContainer – ShoppingBasket – QUANTITY**.



- Open context menu and select item **Insert CellVariant**.
- In popup window *New element* select the type of the associated cell variant UI element: **TableStandardCell**.
- Keep the UI element Id unchanged and press button *Finish*.

New Element

Create new element

Select the type of element you want to create. Optionally you can also specify the ID of the element.

Type:

Id:



6. In the *Outline* view select the newly added *TableStandardCell* UI element
RootUIElementContainer – ShoppingBasket – QUANTITY – TableStandardCell.
7. In the *Properties* tab enter value **SOLD** for property *variantKey*.

Properties	
Property	Value
[-] Element Properties [ViewElement]	
cellDesign	standard
hAlign	auto
id	TableStandardCell
variantKey	SOLD

8. Open context menu for UI element **TableStandardCell** and select menu item **Insert Editor.**
9. In popup window *New element* select the type of the associated cell editor UI element: **TextView.**
10. Keep the UI element Id unchanged and press button *Finish.*

11. In the *Outline* view select the newly added *Editor* UI element **RootUIElementContainer – ShoppingBasket – QUANTITY – TableStandardCell - TextView.**
12. In the *Properties* tab enter value **Sold** for property *text*.

Properties	
Property	Value
[-] Element Properties [TextView]	
design	standard
enabled	true
hAlign	auto
id	TextView
layout	native
semanticColor	standard
text	Sold
textDirection	inherit
tooltip	<>

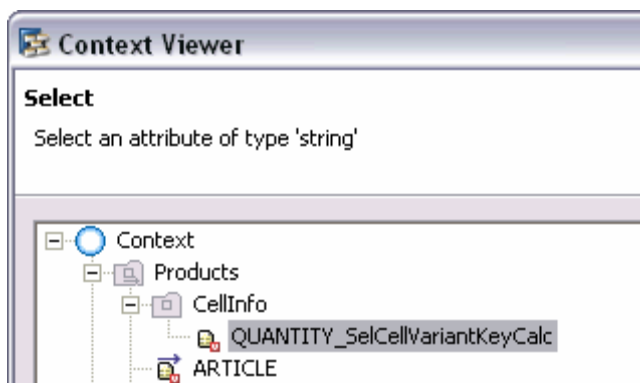
Result

A new cell variant of type *TableStandardCell* is added to the first table column *Quantity*. To address this cell variant a corresponding variant key **SOLD** was set.

STEP 3 – Binding the TableColumn UI Element to the Defined Calculated Context Attribute

In this step we define the data binding relation between table column *QUANTITY* displaying cell variants and the calculated context attribute **QUANTITY_SelCellVariantKeyCal** defined in STEP 1. Based on this binding relation the table retrieves variant keys as Strings for every displayed table row i.e. node element in the table’s data node.

1. In the *Outline* view select the *TableColumn* UI element **RootUIElementContainer – ShoppingBasket – QUANTITY**.
2. In the *Properties* tab select property *selectedCellVariant* which is currently undefined.
3. Press button **...** in column *Value* to define a data binding relation.
4. Select calculated context attribute **Context – Products – CellInfo – QUANTITY_SelCellVariantKeyCalc** and press *OK* button.



Result

The *TableColumn* UI element QUANTITY comprising a table cell variant is bound to the calculated context attribute **QUANTITY_SelCellVariantKeyCalc**. The related getter method calculates the variant key of the cell variant to be displayed for every table row i.e. node element.

Property	Value
Element Properties [TableColumn]	
accessibilityDescription	
cellDesign	standard
design	transparent
filterValue	
fixedPosition	notFixed
groupingValue	
hAlign	auto
id	QUANTITY
isFiltered	false
resizable	true
selectedCellVariant	Products.CellInfo.QUANTITY_SelCellVariantKeyCalc
sortState	none
visible	visible

STEP 4 – Implementing a Calculated Context Attribute Getter Returning Cell Variant Key


Finally we look at the implementation of the calculated context attribute getter method **getCellInfoQUANTITY_SelCellVariantKeyCalc()**. It returns a variant key for every node element in the table's data node.

1. Open the *Implementation* tab for view controller *TableUIExerciseComponent – Views – TableCellVariantsView*.
2. Read the following lines of code in the calculated context attribute getter method **getCellInfoQUANTITY_SelCellVariantKeyCalc()**:

```
public java.lang.String getCellInfoQUANTITY_SelCellVariantKeyCalc(
    IPrivateTableCellVariantsView.ICellInfoElement element)
{
    /**@begin getCellInfoQUANTITY_SelCellVariantKeyCalc(
    // IPrivateTableCellVariantsView.ICellInfoElement)
    IPrivateTableCellVariantsView.IProductsElement prodEl =
        (IPrivateTableCellVariantsView.IProductsElement) element
        .node().getParentElement();
    return prodEl.getISSOLD()
        ? WDTableCellDesign.BADVALUE_LIGHT
        : WDTableCellDesign.STANDARD;
    /**@end
}
```



Testing First Exercise Application TableCellVariantsExerciseApp

1. To save the metadata of your project, choose  (Save All Metadata) in the toolbar.
2. In the Web Dynpro Explorer, open the node *CE254_WDTableUIs_Exercises – Web Dynpro – Applications – TableCellVariantsExerciseApp*.
3. In the node's context menu, choose *Deploy new Archive and Run*. The exercise application is then started in a new browser window.
4. In case you have successfully completed the exercise steps, the first table column *Quantity* displays table rows with different cell editors. For all sold natural clothe articles a *TextView* cell variant with text **Sold** is displayed.

Exercise 1 - Using Cell Variants in Web Dynpro Tables

Natural Clothes: Online Shop

Quantity	Article	Color	Price (€)	Total per article (€)
0	jacket	blue	34,60	
Sold	skirt	red	24,95	
0	t-shirt	orange	29,90	
Sold	trousers	black	64,90	
0	top	black	44,90	
0	dress	colored	78,90	
Sold	blouse	white	35,50	
0	jeans	blue	89,90	
Sold	pullover	red	69,00	
0	sweatshirt	green	61,60	

Zeile 1 von 25



Exercise 2 – Sizing and Layouting Table UIs

Estimated time required for completion: 15 min

Related Web Dynpro Project and Application

This Exercise is based on the following Web Dynpro project and view layout:

- **CD254_TableUIs_Exercises:** Web Dynpro project template comprising the Web Dynpro view this exercise is based on.
- **TableSizingApp:** Exercise application for testing widths-specific table UI element properties.

STEP 1 – Preparation: Running the TableSizing Application in the Web Dynpro HTML Client

1. Open the Web Dynpro project **CD254_TableUIs_Exercises** within the Web Dynpro Explorer.
2. Navigate to node *Web Dynpro – Applications – TableSizingExerciseApp*
3. In the node's context menu, choose **Run**. The exercise application is then started in a new browser window.
4. With this Web Dynpro application You can interactively change and test width-specific table property settings.

Exercise 2 - Sizing Web Dynpro Table Widths

Hide Properties

Table Container

Width:

Height:

Scrolling Mode:

Table Table property settings

Width:

Is Layout Fixed: ReadOnly:

Header Text: Header Visibility:

Design: Grid Mode:

Column 1	Column 2	Column 3
Width: <input type="text"/>	Width: <input type="text"/>	Width: <input type="text"/>
Header Text: <input type="text" value="Column1 Header"/>	Header Text: <input type="text" value="Column2 Header"/>	Header Text: <input type="text" value="Column3 Header"/>
Header Visibility: <input type="text" value="VISIBLE"/>	Header Visibility: <input type="text" value="VISIBLE"/>	Header Visibility: <input type="text" value="VISIBLE"/>
Text Wrapping: <input type="checkbox"/>	Text Wrapping: <input type="checkbox"/>	Text Wrapping: <input type="checkbox"/>

Table Header Table UI

Column1 Header	Column2 Header	Column3 Header
Column 1; Element 0	Column 2; Element 0	Column 3; Element 0
Column 1; Element 1	Column 2; Element 1	Column 3; Element 1
Column 1; Element 2	Column 2; Element 2	Column 3; Element 2
Column 1; Element 3	Column 2; Element 3	Column 3; Element 3
Column 1; Element 4	Column 2; Element 4	Column 3; Element 4

Row 1 of 10



STEP 2 – Setting Absolute Table Widths

1. Set all three table columns to absolute widths: 100px, 100px, 100px. Then change one column width to **50px**. What happens?
 - a. Column width gets smaller
 - b. Column width remains the same
2. Then additionally define an absolute table width by entering width size **300px**. First, set all table column widths to 100px. What happens?
 - a. The table width is equal to the sum of all table column widths. Consequently the table width does not change.
 - b. The table width is equal the sum of all table column widths plus the width of the selection column. Consequently the table width must shrink.
3. Set an absolute column width to 2000px. What's the result?
 - a. _____
4. In the first editable table column enter some very long text. Then set the table's *readOnly* property to **true**. What happens?
 - a. The total table width does not change
 - b. The total table width gets larger as a longer cell text entry must be displayed so that the defined minimal width is not large enough.

STEP 3 – Setting Relative Table Widths

1. First restart the *TableSizingApplication* so that the initial table setting are displayed
2. Set all three table columns to relative widths: 20%, 30%, 50%. Set the table width to 50%. Press Return and look at the displayed table layout. Now set the 50% table column width to 10%. What happens?
 - c. All table columns are resized.
 - d. The width of the table selection column gets larger to fill the missing width (50%) of all three table columns
3. Set all three table columns to relative widths: 30%, 30%, 40%. Set the table width to 50%. Press Return and look at the displayed table layout. Now enter a long column header text by appending the same string many times with copy and paste. Then press return. What happens?
 - a. _____

STEP 4 – Setting Fixed Table Layout and Wrapping Texts in Cell Editors

1. First restart the *TableSizingApplication* so that the initial table setting are displayed
2. In the first editable table column enter some very long text. Then set the table's *readOnly* property to **true**. What happens?
 - a. The total table width does not change but the first table column gets wider whereas the second table column gets smaller
 - b. The total table width does not change as well as the first table column width. The invisible text gets visible via tooltip information.

STEP 5 – Wrapping Texts in Cell Editors

3. First restart the *TableSizingApplication* so that the initial table setting are displayed
4. Set all three table columns to relative widths: 30%, 30%, 40%. Set the table width to 300px.
 - a. Enable text wrapping for the second table column and look at the result in the rendered table UI:



Exercise 3 – Adding Grouped Table Columns

This third exercise dealing with Web Dynpro Tables in SAP NetWeaver 7.0 is based on an incomplete Web Dynpro Project template. After having finished this exercise the following learning objectives will be fulfilled:

- Adding nested table column groups (two hierarchy levels) to a *Table* UI element.
- Adding table columns to existing table column groups with cut & paste.

Estimated time required for completion: 15 min

Related Web Dynpro Project and View Layout

This Exercise is based on the following Web Dynpro project and view layout:

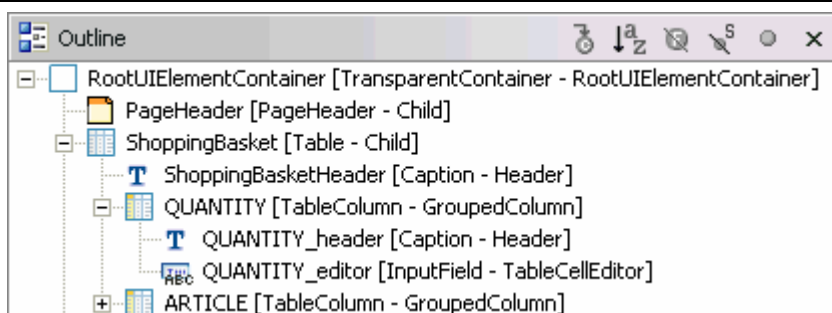
- **CD254_TableUIs_Exercises:** Web Dynpro project template comprising the initial state of the Web Dynpro views this exercises are based on.
- **TableColumnGroupsView:** Exercise view for adding grouped columns to a Web Dynpro table.

The exercise solution can be found in separate Web Dynpro project **CD254_TableUIs**.

STEP 1 – Preparation: Adding Table Columns to a Table in a Flat Hierarchy

Before You can add hierarchically nested table column groups you must first add all table columns in a flat hierarchy directly under the embedding *Table* UI element. After having completed the table binding definitions between table, table columns, table cell editors and the context You can re-arrange all columns in a hierarchically grouped manner.

1. Open the Web Dynpro project **CD254_TableUIs_Exercises** within the Web Dynpro Explorer.
2. Navigate to tree node *Web Dynpro – Web Dynpro Components – TableUIExerciseComponent – Views – TableColumnGroups*
3. Look at the predefined table columns QUANTITY, ARTICLE, COLOR, TOTAL_PER_ARTICLE and PRICE. All columns comprise header and cell editor UI elements and the context binding is already defined. We will now group the existing table columns in a nested hierarchy.

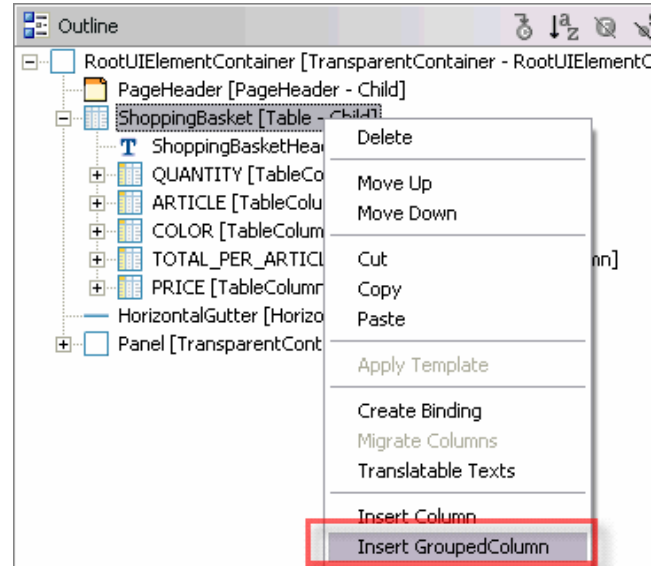


STEP 2 – Adding Nested Table Column Groups To a Table UI Element – Level 1

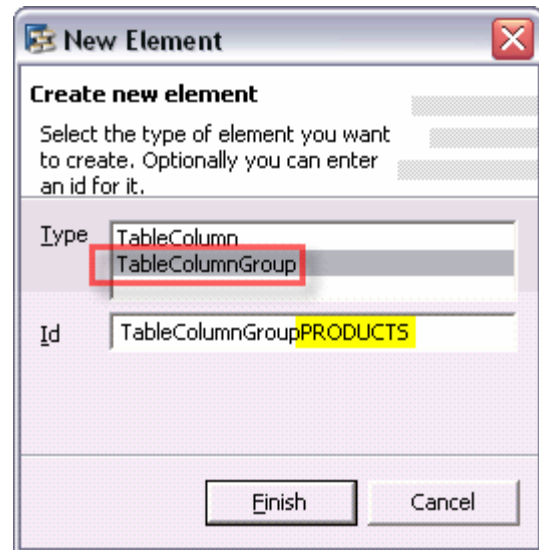
In this step we add a first column group to the existing shopping basket table: The first column group spans all columns with the header text *Products*.



1. In the *Outline* view select *TableColumn* UI element **RootUIElementContainer – ShoppingBasket**.
2. Open context menu for UI element **ShoppingBasket** and select menu item **Insert GroupedColumn**.



3. In popup window *New element* select the type of the table UI element association to be added: **TableColumnGroup**.
4. Enter the UI element Id **TableColumnGroupPRODUCTS** and press button *Finish*.



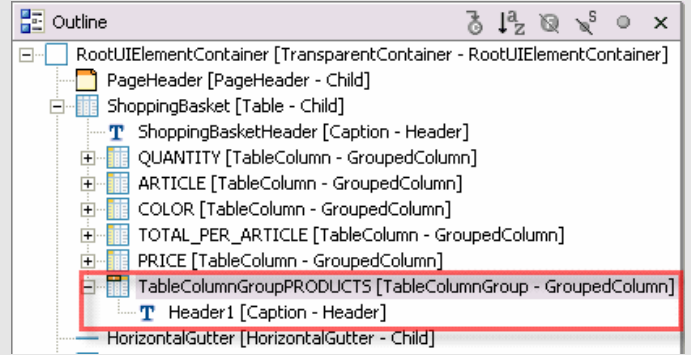
5. In the *Outline* view select *TableColumn* UI element **RootUIElementContainer – ShoppingBasket – TableColumnGroupPRODUCTS**.
6. Select context menu item **Insert Header**.
7. Set property *text* of the newly added header UI element (Caption) to **Products**

Properties	
Property	Value
Element Properties [Caption]	
enabled	true
id	Header1
imageAlt	
imageFirst	true
imageSource	<>
text	Products
textDirection	inherit
tooltip	<>
visible	visible



Result

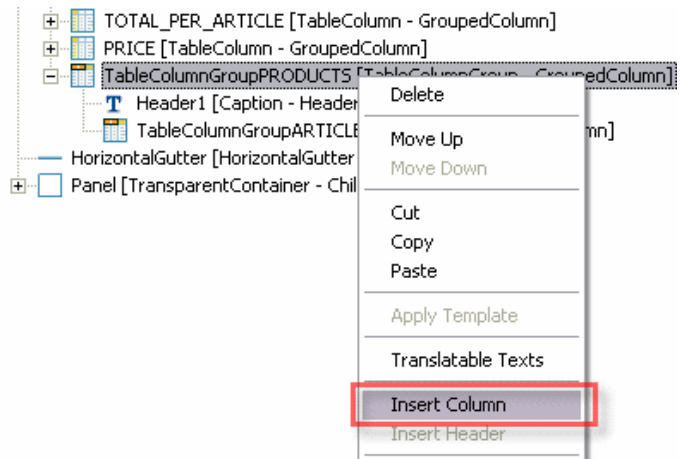
The first *TableColumnGroup* UI element with an associated header is added to the table.



STEP 3 – Adding Nested Table Column Groups To a Table UI Element – Level 2

In this step we add two additional column groups to the existing shopping basket table on a deeper hierarchy level. On the second hierarchy level two column group headers for separating price-related columns (column group header text *Price*) from other article property columns (column group header text *Price*) are added.

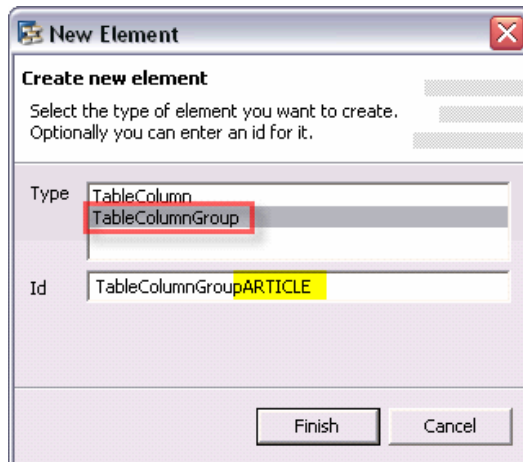
1. In the *Outline* view select *TableColumnGroup* UI element **RootUIElementContainer – ShoppingBasket - TableColumnGroupPRODUCTS**.
2. Open context menu for UI element **TableColumnGroupPRODUCTS** and select menu item **Insert Column**.



NOTE: The UI element association name on deeper table column group hierarchy levels differ from the name on the first hierarchy level.

- first level association name = *GroupedColumn*
- deeper level association name = *Column*

3. In popup window *New element* select the type of the table UI element association to be added: **TableColumnGroup**.
4. Enter the UI element Id **TableColumnGroupARTICLE** and press button *Finish*.





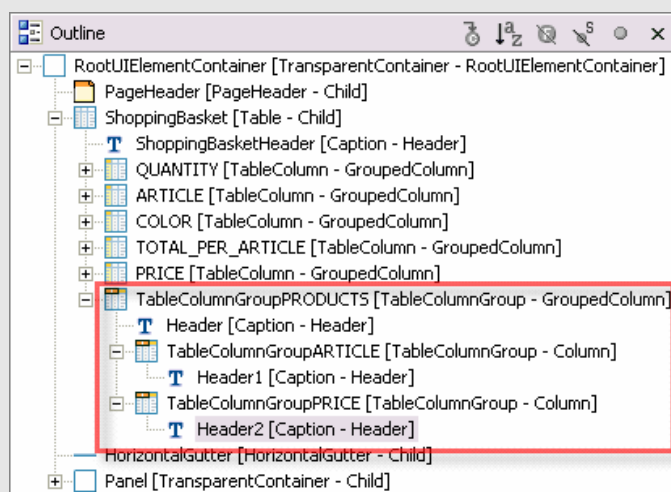
5. In the *Outline* view select *TableColumn* UI element **RootUIElementContainer – ShoppingBasket – TableColumnGroupARTICLE**.
6. Select context menu item **Insert Header**.
7. Set property *text* of the newly added header UI element (Caption) to **Article Details**

Properties	
Property	Value
[-] Element Properties [Caption]	
enabled	true
id	TableColumnGroupARTICLE_Header
imageAlt	
imageFirst	true
imageSource	<>
text	Article Details
textDirection	inherit
tooltip	<>
visible	visible

8. Repeat steps 1-7 adding another *TableColumnGroup* UI element with name **TableColumnGroupPRICE** and header text **Price Information** or apply copy&paste.

Result

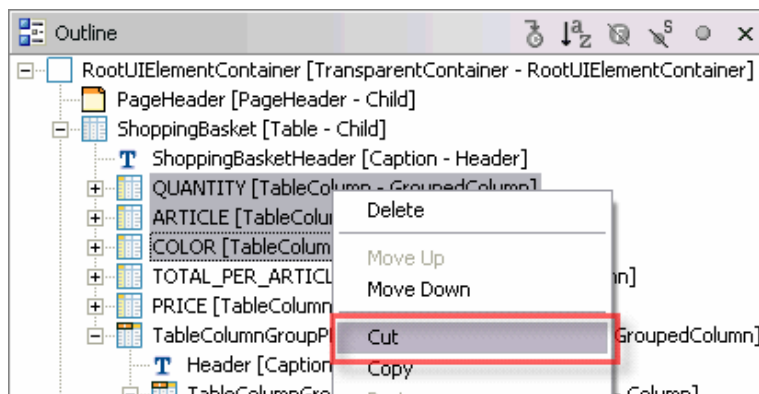
The second level hierarchy comprising two *TableColumnGroup* UI elements with associated headers is added to the table.



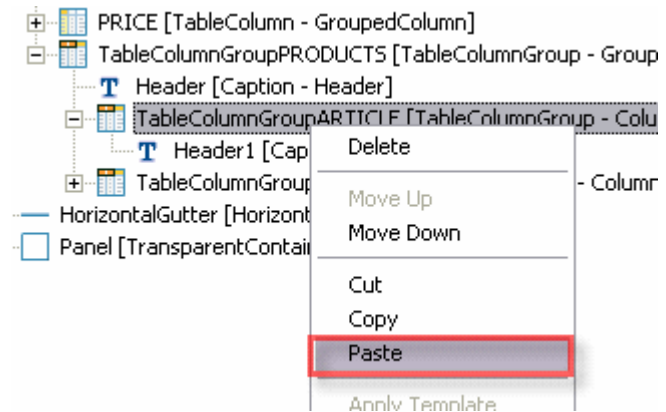
STEP 4 – Adding Table Columns to Table Column Groups With Cut & Paste

In the last exercise step we add the existing table columns to the newly embedded *TableColumnGroupUI* using the **Cut & Paste** function.

1. In the *Outline* view select the three *TableColumn* UI elements **QUANTITY, ARTICLE and COLOR**.
2. Open context menu item **Cut**.



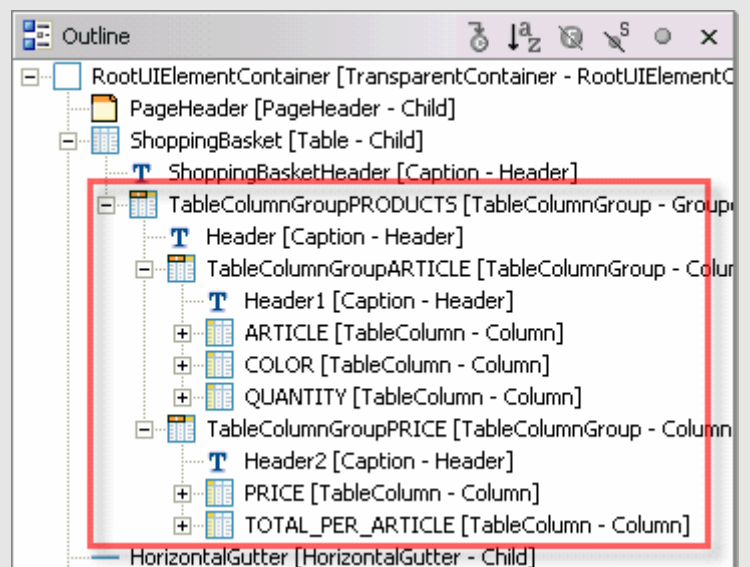
- Paste table columns to the **TableColumnGroupARTICLE** UI element with context menu item *Paste*.



- Repeat steps 1 to 3 with *TableColumn* UI Elements **TOTAL_PER_ARTICLE**, **PRICE** and **TableColumnGroupPRICE**


Result

The second level hierarchy comprising two *TableColumnGroup* UI elements with associated headers is added to the table.





Testing Third Exercise Application TableColumnGroupsExerciseApp

1. To save the metadata of your project, choose  (Save All Metadata) in the toolbar.
2. In the Web Dynpro Explorer, open the node *CE254_WDTableUIs_Exercises – Web Dynpro – Applications – TableColumnGroupsExerciseApp*.
3. In the node's context menu, choose *Deploy new Archive and Run*. The exercise application is then started in a new browser window.
4. In case you have successfully completed the exercise steps, the defined hierarchical table column headers *Products*, *Article* and *Price* are displayed on top of the table columns:

Hierarchical Column Headers					
Natural Clothes: Online Shop					
Products					
Article Details			Price Information		
Quantity	Article	Color	Total per article (€)	Price (€)	
0	jacket	blue	0,00	34,60	
0	skirt	red	0,00	24,95	
0	t-shirt	orange	0,00	29,90	
0	trousers	black	0,00	64,90	



Appendix

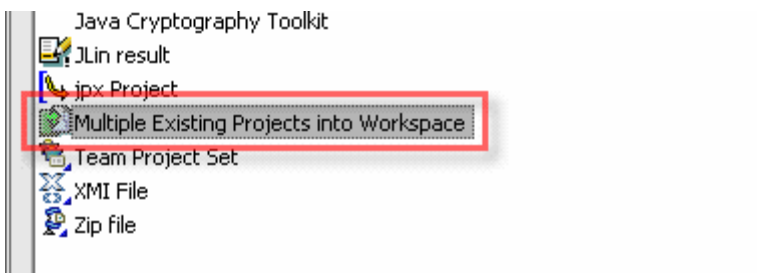
Opening Exercise Web Dynpro Projects in SAP NetWeaver Developer Studio

1. Extract ZIP file [CD254_WDTableUIs_Projects.zip](#) to your **desktop**.

2. Open Your SAP NetWeaver 7.0 Developer Studio

3. Select menu item **File – Import ...**

4. In “*Import*” dialog window select node “*Multiple Existing Projects into Workspace*” and click “*Next*”.



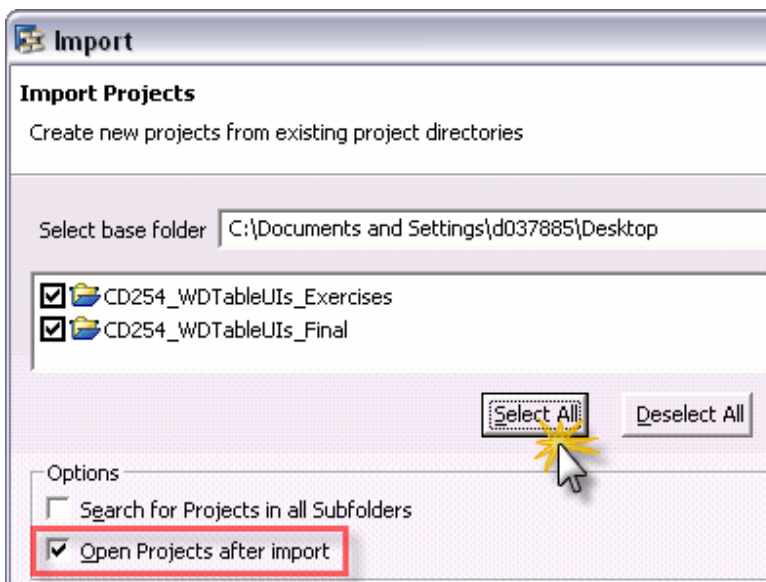
5. In the next dialog window press button “*Browse ...*”

6. In the “*Browse for Folder*” popup window the desktop folder is already selected comprising two Web Dynpro projects to be imported.

7. Press button “*Next*”.

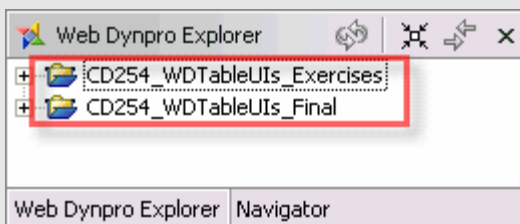
8. In the next dialog window press button “*Select All*” and mark check box “*Open Projects after import*”.

9. Press button “*Finish*”



Result

All required Web Dynpro projects for this hands-on session (Web Dynpro Table UIs in SAP NetWeaver 7.0) are now listed inside the Explorer of the Web Dynpro perspective.





Copyright 2007 SAP AG. All Rights Reserved

- No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.
 - Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.
 - Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.
 - IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, POWER5+, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.
 - Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.
 - Oracle is a registered trademark of Oracle Corporation.
 - UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.
 - Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.
 - HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
 - Java is a registered trademark of Sun Microsystems, Inc.
 - JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
 - MaxDB is a trademark of MySQL AB, Sweden.
 - SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.
-
- The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG.
 - This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document is subject to change and may be changed by SAP at any time without notice.
 - SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.
 - SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.
 - The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.

SAP assumes no responsibility for errors or omissions in these materials